

USER GUIDE

Essential Studio

for EJ2 ASP.NET MVC

Version - v24.1.41 | Release Date - December 18, 2023

| | |
|---|-----|
| Kanban | 89 |
| Getting Started with ASP.NET MVC Kanban Control | 89 |
| Prerequisites | 89 |
| Create ASP.NET MVC application with HTML helper | 89 |
| Install ASP.NET MVC package in the application | 89 |
| Add namespace..... | 89 |
| Add stylesheet and script resources | 89 |
| Register Syncfusion script manager | 90 |
| Add ASP.NET MVC Kanban control | 90 |
| Populating cards..... | 90 |
| Enable swimlane | 94 |
| See also | 95 |
| Columns in Kanban Board | 95 |
| Single-key mapping | 95 |
| Multi-key mapping | 99 |
| Header text | 103 |
| Header template | 103 |
| Toggle columns | 109 |
| Stacked headers | 117 |
| Cards in ASP.NET MVC Kanban component..... | 121 |
| Drag-and-drop..... | 121 |
| Header..... | 121 |
| Content | 125 |
| Template | 125 |
| Selection..... | 130 |
| Data binding in ASP.NET MVC Kanban component | 134 |
| Local data | 134 |
| Remote data..... | 138 |
| Loading data via ajax..... | 147 |
| Swimlane in ASP.NET MVC Kanban control | 148 |
| Render swimlane row | 148 |
| Custom row text..... | 152 |
| Template | 156 |
| Sorting | 160 |
| Drag-and-drop..... | 164 |

| | |
|---|-----|
| Create empty row | 167 |
| Calculate cards count..... | 171 |
| Enable frozen rows | 175 |
| Drag and drop in ASP.NET MVC Kanban control | 179 |
| Internal drag and drop | 179 |
| External drag and drop | 186 |
| Sort in ASP.NET MVC Kanban control | 200 |
| Index..... | 200 |
| DataSource Order | 208 |
| Custom | 211 |
| Change the direction..... | 214 |
| Card Editing in ASP.NET MVC Kanban control | 218 |
| Default Dialog..... | 218 |
| Custom Fields..... | 222 |
| Dialog Template | 235 |
| Prevent Dialog..... | 241 |
| Persisting data in server..... | 244 |
| Tooltip in ASP.NET MVC Kanban control | 252 |
| Tooltip template | 252 |
| Card Order in ASP.NET MVC Kanban control..... | 256 |
| Validation in ASP.NET MVC Kanban control | 261 |
| Minimum card limit..... | 261 |
| Maximum card limit | 261 |
| Virtualization | 265 |
| Virtual scrolling | 265 |
| Limitations for virtual scrolling | 272 |
| Globalization in ASP.NET MVC Kanban control | 272 |
| Loading translations..... | 273 |
| Right to left (RTL) | 277 |
| Kanban dimensions in ASP.NET MVC Kanban control | 281 |
| Auto height and width | 281 |
| Height and width in pixel | 285 |
| Height and width in percentage | 289 |
| State Persistence in ASP.NET MVC Kanban control..... | 293 |
| Responsive mode | 296 |

| | |
|---|-----|
| Layouts | 296 |
| Scrolling | 298 |
| Selection | 299 |
| Styling and appearance | 302 |
| To set fixed position to the Kanban header | 305 |
| Accessibility in ASP.NET MVC Kanban control | 305 |
| WAI-ARIA attributes | 306 |
| Keyboard interaction | 307 |
| Disable keyboard interaction | 307 |
| Ensuring accessibility | 311 |
| See also | 311 |
| How To | 311 |
| Add header double click | 311 |
| Change Kanban columns dynamically | 315 |
| Filtering Cards | 319 |
| Searching Cards | 323 |
| Linear Gauge | 327 |
| Getting Started with ASP.NET MVC Linear Gauge Control | 327 |
| Prerequisites | 327 |
| Create ASP.NET MVC application with HTML helper | 328 |
| Install ASP.NET MVC package in the application | 328 |
| Add namespace | 328 |
| Add script resources | 328 |
| Register Syncfusion script manager | 328 |
| Add ASP.NET MVC Linear Gauge control | 329 |
| Add Gauge Title | 329 |
| Axis | 330 |
| Dimensions in ASP.NET MVC Linear Gauge | 331 |
| Size for Linear Gauge | 331 |
| Axis in ASP.NET MVC Linear Gauge | 333 |
| Setting the start value and end value of the axis | 333 |
| Line Customization | 334 |
| Ticks Customization | 335 |
| Labels Customization | 338 |
| Orientation | 346 |

| | |
|---|-----|
| Inverted Axis | 347 |
| Opposed Axis..... | 348 |
| Multiple Axes | 349 |
| Ranges in ASP.NET MVC Linear Gauge..... | 351 |
| Customizing the range | 352 |
| Setting the range color for the labels | 354 |
| Multiple ranges | 355 |
| Gradient Color..... | 357 |
| Radial Gradient | 359 |
| Pointers in ASP.NET MVC Linear Gauge..... | 360 |
| Types of pointer | 362 |
| Multiple pointers | 371 |
| Pointer animation | 372 |
| Gradient Color..... | 373 |
| Annotations in ASP.NET MVC Linear Gauge | 377 |
| Adding annotation | 377 |
| Customization | 378 |
| Multiple annotations | 383 |
| Animation in ASP.NET MVC Linear Gauge | 384 |
| User Interaction in ASP.NET Linear Gauge..... | 386 |
| Tooltip | 386 |
| Positioning the tooltip..... | 391 |
| Pointer Drag | 393 |
| Print and Export in ASP.NET MVC Linear Gauge..... | 394 |
| Print..... | 394 |
| Export..... | 395 |
| Appearance in ASP.NET MVC Linear Gauge..... | 399 |
| Customizing the Linear Gauge area | 399 |
| Setting up the Linear Gauge title | 400 |
| Customizing the Linear Gauge container | 402 |
| Fitting the Linear Gauge to the control..... | 407 |
| Accessibility in ASP.NET MVC Linear Gauge | 408 |
| WAI-ARIA attributes..... | 408 |
| Screen reading in Linear Gauge | 408 |
| Ensuring accessibility | 409 |

| | |
|--|-----|
| See also | 409 |
| Internationalization in ASP.NET MVC Linear Gauge | 409 |
| Numeric Format | 409 |
| Events in ASP.NET MVC Linear Gauge | 410 |
| AnimationComplete | 410 |
| AnnotationRender | 411 |
| AxisLabelRender | 411 |
| BeforePrint | 411 |
| DragEnd | 412 |
| DragMove | 412 |
| DragStart | 413 |
| GaugeMouseDown | 413 |
| GaugeMouseLeave | 413 |
| GaugeMouseMove | 414 |
| GaugeMouseUp | 414 |
| Load | 414 |
| Loaded | 414 |
| Resized | 415 |
| TooltipRender | 415 |
| ValueChange | 415 |
| Methods in ASP.NET MVC Linear Gauge | 416 |
| setPointerValue | 416 |
| setAnnotationValue | 417 |
| refresh | 417 |
| Migration from Essential JS 1 | 418 |
| Linear gauge dimensions | 418 |
| Line customization | 418 |
| Ticks | 419 |
| Labels | 420 |
| Axis | 422 |
| Ranges | 422 |
| Bar Pointer | 423 |
| Marker Pointer | 424 |
| Annotation | 426 |
| Tooltip | 427 |

| | |
|--|-----|
| Appearance of Linear Gauge..... | 428 |
| Gauge Container type | 428 |
| Events..... | 429 |
| How To | 430 |
| Rendering linear gauges dynamically | 431 |
| ListBox..... | 437 |
| Getting Started with ASP.NET MVC ListBox Control | 437 |
| Prerequisites | 437 |
| Create ASP.NET MVC application with HTML helper..... | 437 |
| Install ASP.NET MVC package in the application | 437 |
| Add namespace..... | 438 |
| Add stylesheet and script resources | 438 |
| Register Syncfusion script manager | 438 |
| Add ASP.NET MVC ListBox control..... | 439 |
| Accessibility in List Box Component..... | 439 |
| WAI-ARIA attributes..... | 440 |
| Keyboard interaction | 441 |
| Ensuring accessibility | 441 |
| See also | 441 |
| Data Binding in ASP.NET MVC List Box Control | 441 |
| Local Data..... | 442 |
| Remote Data | 444 |
| Drag and drop in ListBox Control | 444 |
| Single listbox | 445 |
| Multiple listbox | 445 |
| Dual list box in ASP.NET MVC ListBox Control | 446 |
| Icons and Customization | 447 |
| Icons | 447 |
| Templates..... | 451 |
| Selection in ASP.NET MVC ListBox Control | 452 |
| Single selection | 453 |
| Multiple selection | 453 |
| sorting and grouping..... | 454 |
| Sorting..... | 454 |
| Grouping | 454 |

| | |
|---|-----|
| Styles and Appearances | 456 |
| Horizontal ListBox | 456 |
| How To | 457 |
| Add items to the list box | 457 |
| Enable or disable items | 458 |
| Enable Scroller | 458 |
| Form submit to the list box | 459 |
| Select items to the list box | 460 |
| ListView | 460 |
| Getting Started with ASP.NET MVC ListView control | 460 |
| Prerequisites | 460 |
| Create ASP.NET MVC application with HTML helper | 461 |
| Install ASP.NET MVC package in the application | 461 |
| Add namespace | 461 |
| Add stylesheet and script resources | 461 |
| Register Syncfusion script manager | 462 |
| Add ASP.NET MVC ListView control | 462 |
| Bind dataSource | 462 |
| See also | 463 |
| Data binding | 463 |
| Bind to local data | 464 |
| Bind to remote data | 465 |
| Grouping | 466 |
| Customization | 468 |
| Checklist in ASP.NET MVC Listview Control | 468 |
| Checkbox Position | 470 |
| Nested List | 472 |
| Customizing templates | 475 |
| Header Template | 475 |
| Template | 477 |
| Group template | 481 |
| UI Virtualization | 483 |
| Getting started | 483 |
| Conditional rendering | 488 |
| Scrolling in ASP.NET MVC Listview Control | 490 |

| | |
|--|-----|
| CSS Structure..... | 493 |
| Customizing ListView | 493 |
| Customizing the list items | 494 |
| Customizing ListView's header..... | 494 |
| Customizing group header of ListView | 494 |
| Customizing the hover state of ListView control | 494 |
| Customizing selected item of ListView control..... | 495 |
| Accessibility in ASP.NET MVC ListView component..... | 495 |
| WAI-ARIA attributes..... | 496 |
| Keyboard interaction | 497 |
| Ensuring accessibility | 497 |
| See also | 497 |
| Migration from Essential JS 1..... | 497 |
| How To | 500 |
| Get selected items from ListView | 500 |
| Use Dynamic templates in ListView based on device | 502 |
| Create Dual List from ListView | 507 |
| Load List Items in child list dynamically | 514 |
| Show Items Count in Group Header | 516 |
| Filter List Items in the ListView | 519 |
| Add and Remove List Items from the ListView | 521 |
| Trace all events in ListView | 523 |
| Create Mobile contact layout from ListView | 526 |
| Display spinner until List Items are loaded | 530 |
| Hide checkbox in ListView..... | 532 |
| Manipulate ListView as Grid Layout | 536 |
| element .e-list-item { | 536 |
| Drag and Drop List Items..... | 543 |
| Render ListView with hyper-link navigation | 545 |
| Customize ListView as Chat Window | 547 |
| Customize Each List Item's Text in ListView Component..... | 552 |
| Maps | 553 |
| Getting Started with ASP.NET MVC Maps Component | 553 |
| Prerequisites | 553 |
| Create ASP.NET MVC application with HTML helper | 553 |

| | |
|--|-----|
| Install ASP.NET MVC package in the application | 553 |
| Add namespace..... | 554 |
| Add script resources | 554 |
| Register Syncfusion script manager | 554 |
| Add ASP.NET MVC Maps component | 554 |
| Populate data..... | 555 |
| Geometry types..... | 555 |
| Shape data | 556 |
| Data source | 556 |
| Data binding..... | 557 |
| Binding complex data source | 560 |
| Layers | 562 |
| Multilayer..... | 562 |
| Sublayer | 562 |
| Displaying different layer in the view | 564 |
| Rendering custom shapes | 566 |
| Providers | 566 |
| OpenStreetMaps in ASP.NET MVC Maps Component..... | 566 |
| Bing Maps in ASP.NET MVC Maps Component..... | 575 |
| Azure Maps in ASP.NET MVC Maps Component | 587 |
| Other Maps providers | 595 |
| Customization in ASP.NET MVC Maps Component | 605 |
| Setting the size for Maps | 605 |
| Maps title | 607 |
| Setting theme..... | 609 |
| Customizing Maps container | 611 |
| Customizing Maps area..... | 613 |
| Customizing the shapes | 615 |
| Setting color to the shapes from the data source | 617 |
| Applying border to individual shapes | 619 |
| Projection type..... | 621 |
| Color Mapping in ASP.NET MVC Maps Component | 623 |
| Types of color mapping..... | 624 |
| Multiple colors for a single shape | 631 |
| Color for items excluded from color mapping | 633 |

| | |
|--|-----|
| Color mapping for bubbles | 636 |
| Data labels in ASP.NET MVC Maps Component..... | 637 |
| Adding data labels..... | 638 |
| Customization | 641 |
| Label animation..... | 643 |
| Smart labels..... | 644 |
| Intersect action | 646 |
| Adding data label as a template | 648 |
| Polygon shape in ASP.NET MVC Maps component | 650 |
| Markers | 653 |
| Adding marker..... | 653 |
| Adding marker template | 655 |
| Customization | 657 |
| Marker shapes | 659 |
| Multiple marker groups | 662 |
| Customize marker shapes from data source | 664 |
| Repositioning the marker using drag and drop | 668 |
| Marker zooming..... | 670 |
| Marker clustering..... | 672 |
| Customization of marker cluster..... | 675 |
| Expanding the marker cluster | 677 |
| Tooltip for marker | 680 |
| Bubbles in ASP.NET MVC Maps Component | 682 |
| Bubble shapes | 684 |
| Customization | 686 |
| Setting colors to the bubbles from the data source | 688 |
| Setting the range of the bubble size | 690 |
| Multiple bubble groups..... | 692 |
| Enable tooltip for bubble | 695 |
| Legend in ASP.NET MVC Maps Component..... | 696 |
| Modes of legend | 696 |
| Positioning of the legend | 699 |
| Legend for shapes | 701 |
| Enable legend for bubbles | 721 |
| Enable legend for markers | 722 |

| | |
|---|-----|
| Annotations in ASP.NET MVC Maps Component..... | 726 |
| Annotation | 726 |
| Annotation customization | 728 |
| Multiple Annotation..... | 732 |
| Navigation lines..... | 734 |
| Customization | 734 |
| Enabling the arrows | 736 |
| User Interactions..... | 738 |
| Zooming | 738 |
| Selection..... | 752 |
| Highlight | 768 |
| Tooltip | 777 |
| Print and Export in ASP.NET MVC Maps component..... | 785 |
| Print..... | 785 |
| Export..... | 786 |
| State Persistence..... | 790 |
| State persistence..... | 790 |
| Accessibility in ASP.NET MVC Maps component | 792 |
| WAI-ARIA attributes..... | 792 |
| Screen reading in Maps..... | 792 |
| Keyboard Navigation..... | 793 |
| Ensuring accessibility | 793 |
| See also | 793 |
| Internationalization..... | 793 |
| Globalization | 794 |
| Numeric format..... | 794 |
| Localization | 796 |
| Methods in ASP.NET MVC Maps component | 798 |
| Methods | 798 |
| getMinMaxLatitudeLongitude | 798 |
| Migration from Essential JS 1..... | 800 |
| Size Customization | 800 |
| Title and Subtitle Customization | 801 |
| Layer Customization..... | 801 |
| Shape Customization | 802 |

| | |
|--|-----|
| Marker Customization | 803 |
| Bubble Customization | 805 |
| DataLabel Customization | 807 |
| Legend Customization..... | 808 |
| Zooming Customization | 811 |
| Highlight And Selection Customization..... | 812 |
| Navigation Line Customization | 814 |
| Tooltip Customization | 815 |
| Annotation Cutomization..... | 816 |
| Maps Other Properties Customization | 816 |
| Events..... | 817 |
| How To | 820 |
| Annotations in ASP.NET MVC Maps Component..... | 820 |
| Custom path map in ASP.NET MVC Maps Component..... | 822 |
| Drill-down in ASP.NET MVC Maps Component | 824 |
| Marker types in ASP.NET MVC Maps Component..... | 827 |
| Adding multiple layers in the Map..... | 832 |
| Center position zooming..... | 834 |
| MaskedTextBox..... | 836 |
| Getting Started with ASP.NET MVC MaskedTextBox control | 836 |
| Prerequisites | 836 |
| Create ASP.NET MVC application with HTML helper..... | 836 |
| Install ASP.NET MVC package in the application | 836 |
| Add namespace..... | 836 |
| Add stylesheet and script resources | 837 |
| Register Syncfusion script manager..... | 837 |
| Add ASP.NET MVC MaskedTextBox control..... | 837 |
| Set the mask..... | 838 |
| See also | 838 |
| Mask Configuration in MaskedTextBox Control | 838 |
| Standard mask elements..... | 838 |
| Custom mask elements..... | 840 |
| Prompt character | 842 |
| Accessibility..... | 842 |
| WAI-ARIA attributes..... | 843 |

| | |
|--|-----|
| Ensuring accessibility | 844 |
| See also | 844 |
| Style and appearance in MaskedTextBox Component | 844 |
| Customizing the appearance of MaskedTextBox wrapper element..... | 844 |
| Customizing the MaskedTextBox element on hovering | 844 |
| How To | 845 |
| Perform custom validation using FormValidator | 845 |
| Set cursor position while focus on the input textbox | 846 |
| Display numeric keypad when focus on mobile devices | 847 |
| Customize the UI appearance of the control..... | 848 |
| MaskedTextBoxFor and Model Binding..... | 849 |
| Migration from Essential JS 1..... | 850 |
| Common..... | 850 |
| Mask Configuration..... | 853 |
| Validation | 855 |
| Mention..... | 855 |
| Getting Started with ASP.NET MVC Mention Control | 855 |
| Prerequisites | 855 |
| Create ASP.NET MVC application with HTML helper | 856 |
| Install ASP.NET MVC package in the application | 856 |
| Add namespace..... | 856 |
| Add stylesheet and script resources | 856 |
| Register Syncfusion script manager | 857 |
| Add ASP.NET MVC Mention control | 857 |
| Binding data source | 857 |
| Display mention character | 858 |
| Working with Data in Mention | 859 |
| Binding local data..... | 859 |
| Binding remote data | 864 |
| See also | 866 |
| Filtering data in Mention | 866 |
| Limit the minimum filter character..... | 866 |
| Change the filter type | 868 |
| Allow spacing between search..... | 869 |
| Customize the suggestion item count | 870 |

| | |
|--|-----|
| See also | 872 |
| Sorting datasource in Mention | 872 |
| Sort order type | 872 |
| Templates in Mention..... | 873 |
| Item template | 873 |
| Display template | 875 |
| No records template | 876 |
| Spinner template | 877 |
| See also | 879 |
| Localization in ASP.NET MVC Mention | 879 |
| See also | 879 |
| Customization in Mention | 879 |
| Show or hide mention character | 879 |
| Adding the suffix character after selection..... | 881 |
| Configure the popup list | 882 |
| Trigger character | 883 |
| Accessibility in Mention | 883 |
| WAI-ARIA attributes..... | 884 |
| Keyboard interaction | 885 |
| Ensuring accessibility | 886 |
| See also | 886 |
| Menu..... | 886 |
| Getting Started with ASP.NET MVC Menu Control..... | 886 |
| Prerequisites | 886 |
| Create ASP.NET MVC application with HTML helper..... | 886 |
| Install ASP.NET MVC package in the application | 886 |
| Add namespace..... | 887 |
| Add stylesheet and script resources | 887 |
| Register Syncfusion script manager | 887 |
| Add ASP.NET MVC Menu control..... | 888 |
| Group menu items with separator | 889 |
| See also | 891 |
| Icons and Sub menu items | 891 |
| Icons | 891 |
| Navigation | 894 |

| | |
|--|-----|
| Multilevel nesting | 895 |
| See Also | 898 |
| Data source binding and Custom menu items..... | 898 |
| Data binding..... | 898 |
| Custom menu items | 902 |
| See Also | 905 |
| Use case scenarios | 905 |
| Scrollable menu | 905 |
| Menu in toolbar | 908 |
| Hamburger menu..... | 911 |
| Mobile view..... | 915 |
| Accessibility in Menu Control | 920 |
| WAI-ARIA attributes..... | 921 |
| Keyboard interaction | 921 |
| Ensuring accessibility | 922 |
| See also | 922 |
| Styles and Appearances | 922 |
| How To | 922 |
| Change orientation in Menu Control..... | 922 |
| Customize menu using events | 924 |
| Customize menu items | 926 |
| Create mnemonic UI in menu item..... | 931 |
| Change sub menu position | 934 |
| Menu with rounded corner..... | 936 |
| Set title for Menu Items..... | 938 |
| Open menu and sub menu on click only..... | 939 |
| Migration from Essential JS 1..... | 941 |
| Properties..... | 941 |
| Methods..... | 943 |
| Events..... | 945 |
| Message | 946 |
| Getting Started with ASP.NET MVC Message Control | 946 |
| Prerequisites | 946 |
| Create ASP.NET MVC application with HTML helper | 946 |
| Install ASP.NET MVC package in the application | 946 |

| | |
|--|-----|
| Add namespace..... | 946 |
| Add stylesheet and script resources | 947 |
| Register Syncfusion script manager | 947 |
| Add ASP.NET MVC Message control | 947 |
| Severities in Message control | 948 |
| Variants in Message control | 949 |
| Icons in Message control | 951 |
| No Icon | 952 |
| Custom Icon | 953 |
| Close Icon | 954 |
| Customization in Message control..... | 957 |
| Content Alignment | 957 |
| Rounded and Square..... | 958 |
| CSS Message..... | 959 |
| Template in Message control | 960 |
| Accessibility in ASP.NET MVC Message control | 962 |
| WAI-ARIA attributes..... | 963 |
| Keyboard interaction | 963 |
| Ensuring accessibility | 963 |
| See also | 963 |
| MultiSelect | 963 |
| Getting Started with ASP.NET MVC MultiSelect Control | 963 |
| Prerequisites | 963 |
| Create ASP.NET MVC application with HTML helper | 963 |
| Install ASP.NET MVC package in the application | 963 |
| Add namespace..... | 964 |
| Add stylesheet and script resources | 964 |
| Register Syncfusion script manager | 964 |
| Add ASP.NET MVC MultiSelect control | 965 |
| Configure the popup list | 965 |
| See also | 966 |
| Data Binding..... | 966 |
| Binding local data..... | 966 |
| Binding remote data | 969 |
| See Also | 974 |

| | |
|---|-----|
| Templates in MultiSelect Control | 974 |
| Item template | 974 |
| Value template..... | 975 |
| Group template..... | 976 |
| Header template | 977 |
| Footer template | 978 |
| No records template | 979 |
| Action failure template | 979 |
| See Also | 980 |
| Grouping | 980 |
| Customization | 981 |
| Grouping with CheckBox..... | 981 |
| Filtering | 982 |
| Limit the minimum filter character..... | 983 |
| Change the filter type | 984 |
| Case sensitive filtering | 985 |
| Diacritics Filtering..... | 986 |
| See Also | 987 |
| CustomValue | 987 |
| CheckBox in MultiSelect Control..... | 988 |
| Select All..... | 989 |
| Selection Limit..... | 990 |
| Selection Reordering..... | 990 |
| See Also | 991 |
| Chip Customization | 991 |
| Localization in MultiSelct Control | 993 |
| Loading translations..... | 994 |
| See Also | 994 |
| CSS structures | 995 |
| Customizing the background color of wrapper element | 995 |
| Customizing the appearance of the delimiter wrapper element | 995 |
| Customizing the appearance of chips | 995 |
| Customizing the dropdown icon's color | 995 |
| Customizing the focus color | 996 |
| Customizing the disabled component's text color | 996 |

| | |
|---|------|
| Customizing the color of the placeholder text | 996 |
| Customizing the placeholder to add mandatory indicator(*)..... | 996 |
| Customizing the float label element's focusing color | 997 |
| Customizing the outline theme's focus color | 997 |
| Customizing the background color of focus, hover, and active item's | 997 |
| Customizing the appearance of pop-up element | 998 |
| Customizing the color of the checkbox..... | 998 |
| Accessibility..... | 998 |
| WAI-ARIA attributes..... | 999 |
| Keyboard interaction | 1000 |
| Ensuring accessibility | 1001 |
| See also | 1001 |
| How To | 1001 |
| Show the list items with icons | 1001 |
| Set preselected items through fields | 1005 |
| MultiSelectFor..... | 1006 |
| Migration from Essential JS 1..... | 1009 |
| Accessibility and Localization..... | 1009 |
| Animation..... | 1009 |
| Template | 1009 |
| Data Binding..... | 1010 |
| Filtering | 1011 |
| Popups | 1011 |
| Selection..... | 1013 |
| Common..... | 1014 |
| Numeric TextBox..... | 1018 |
| Getting Started with ASP.NET MVC NumericTextBox Control..... | 1018 |
| Prerequisites | 1018 |
| Create ASP.NET MVC application with HTML helper..... | 1018 |
| Install ASP.NET MVC package in the application | 1019 |
| Add namespace..... | 1019 |
| Add stylesheet and script resources | 1019 |
| Register Syncfusion script manager | 1019 |
| Add ASP.NET MVC NumericTextBox control | 1020 |
| Range validation..... | 1020 |

| | |
|---|------|
| Formatting the value..... | 1020 |
| Precision of numbers | 1020 |
| See also | 1021 |
| Number Formats in NumericTextBox Control | 1021 |
| Standard formats | 1021 |
| Custom formats | 1022 |
| Globalization | 1023 |
| Internationalization..... | 1023 |
| Localization | 1024 |
| Right to Left..... | 1028 |
| Accessibility..... | 1031 |
| WAI-ARIA attributes..... | 1032 |
| Keyboard interaction | 1033 |
| Ensuring accessibility | 1033 |
| See also | 1033 |
| Style and appearance in NumericTextBox Component | 1034 |
| Customizing the appearance of NumericTextBox wrapper element..... | 1034 |
| Customizing the NumericTextBox icons | 1034 |
| How To | 1034 |
| Customize the UI appearance of the control..... | 1034 |
| Customize the spin button's up and down arrow..... | 1035 |
| Customize the step value and hide spin buttons..... | 1036 |
| Maintain trailing zeros in NumericTextBox..... | 1037 |
| NumericTextBoxFor and Model Binding..... | 1037 |
| Perform custom validation using FormValidator | 1039 |
| Prevent nullable input in NumericTextBox | 1040 |
| Migration from Essential JS 1..... | 1041 |
| Common..... | 1041 |
| Globalization | 1043 |
| Group | 1043 |
| Numeric configuration | 1043 |
| Number Formats | 1044 |
| Validation | 1045 |
| PDF Viewer..... | 1045 |
| Getting Started..... | 1045 |

| | |
|---|------|
| Getting Started with ASP.NET MVC Standalone PDF Viewer Control..... | 1045 |
| Getting Started with ASP.NET MVC PDF Viewer Control..... | 1051 |
| Feature modules | 1060 |
| See also | 1061 |
| Open PDF files | 1061 |
| Open PDF file from AWS S3 | 1061 |
| Open PDF file from Azure Blob Storage | 1063 |
| Open PDF file from Google Cloud Storage..... | 1065 |
| Open PDF file from Google Drive | 1067 |
| Open PDF file from Box cloud file storage | 1071 |
| Save PDF files | 1073 |
| Save PDF file to AWS S3 | 1073 |
| Save PDF file to Azure Blob Storage..... | 1075 |
| Save PDF file to Google Cloud Storage | 1077 |
| Save PDF file to Google Drive..... | 1079 |
| Save PDF file to Box cloud file storage..... | 1082 |
| Accessibility in Syncfusion PDF Viewer components | 1084 |
| WAI-ARIA attributes..... | 1085 |
| Keyboard interaction | 1086 |
| Ensuring accessibility | 1087 |
| Built-in toolbar | 1087 |
| Show/Hide the default toolbar | 1088 |
| Show/Hide the default toolbaritem..... | 1088 |
| See also | 1089 |
| Navigation in PDF Viewer component | 1089 |
| Toolbar page navigation option | 1089 |
| Bookmark navigation | 1090 |
| Thumbnail navigation | 1091 |
| Hyperlink navigation | 1092 |
| Table of content navigation | 1092 |
| See also | 1093 |
| Magnification | 1094 |
| See also | 1095 |
| Text Search..... | 1095 |
| See also | 1096 |

| | |
|--|------|
| Annotation | 1096 |
| Text Markup Annotation in the ASP.NET MVC PDF Viewer component | 1096 |
| Shape Annotation in the ASP.NET MVC PDF Viewer component | 1108 |
| Stamp Annotation in the ASP.NET MVC PDF Viewer component | 1113 |
| Sticky Notes Annotation in the ASP.NET MVC PDF Viewer component | 1116 |
| Measurement Annotation in the ASP.NET MVC PDF Viewer component | 1120 |
| Free Text Annotation in the ASP.NET MVC PDF Viewer component | 1127 |
| Comments in the ASP.NET MVC PDF Viewer component | 1133 |
| Ink Annotation in the ASP.NET MVC PDF Viewer component | 1137 |
| Form Filling | 1140 |
| Disabling form fields | 1140 |
| How to draw handwritten signature in the signature field | 1141 |
| Delete the signature inside the signature field | 1141 |
| Import and export form fields | 1141 |
| Importing form fields using PDF Viewer API | 1141 |
| Exporting form fields from the PDF document using PDF Viewer API | 1142 |
| Handwritten Signature | 1142 |
| Adding a handwritten signature to the PDF document | 1143 |
| Editing the properties of handwritten signature | 1144 |
| See also | 1145 |
| Interaction Mode in PDF Viewer component | 1145 |
| Selection mode | 1145 |
| Panning Mode | 1146 |
| See also | 1147 |
| Form Designer | 1147 |
| Create form fields programmatically | 1147 |
| Create form fields with UI interaction | 1178 |
| Print | 1183 |
| See also | 1184 |
| Download a PDF document in PDF Viewer component | 1184 |
| See also | 1185 |
| Localization | 1185 |
| Server Actions with ASP.NET MVC PDFViewer Control | 1190 |
| Load action | 1190 |
| RenderPdfPages | 1192 |

| | |
|---|------|
| RenderThumbnaiImages action | 1192 |
| Bookmarks | 1193 |
| RenderAnnotationComments | 1193 |
| Unload action | 1193 |
| ExportAnnotations action | 1194 |
| ImportAnnotations..... | 1194 |
| ImportFormFields action..... | 1195 |
| ExportFormFields action | 1196 |
| Download action | 1196 |
| PrintImages | 1196 |
| Other methods..... | 1197 |
| jsonObjects class | 1198 |
| How To | 1200 |
| Customize the toolbar..... | 1200 |
| pdfviewer { | 1202 |
| magnificationToolbar { | 1202 |
| magnificationToolbar.e-toolbar .e-tbar-btn { | 1202 |
| topToolbar { | 1203 |
| Exporting PDFs as raster images | 1210 |
| Extract Text from PDF document..... | 1213 |
| Delete a specific annotation using deleteAnnotationById | 1214 |
| Unload the PDF document programmatically | 1214 |
| Set author name to annotation | 1215 |
| Identify the loaded document is edited..... | 1215 |
| Close the comment panel | 1216 |
| Access file name..... | 1216 |
| Include Authorization token | 1217 |
| Achieve Load balancing environment in MVC framework | 1218 |
| Extract Text using TextLineCollection | 1221 |
| Load the PDF document..... | 1222 |
| Show the notification dialog in UI When form fields are empty | 1223 |
| Load PDF documents dynamically | 1224 |
| Import and Export annotation as object..... | 1225 |
| Open Thumbnail pane programmatically | 1226 |
| Redirect to a home page after submitting PDF forms | 1227 |

| | |
|---|------|
| pdfviewer { | 1227 |
| Resolve the Pdfium issue | 1238 |
| How to add the date to the signature text | 1240 |
| How to change the default width and height | 1241 |
| Resolve "Unable to find an entry point named FPDFText_GetCharAngle" error | 1241 |
| How to clear the "Web-service is not listening" to error | 1242 |
| Load N number of pages on initial loading | 1244 |
| Retry Timeout | 1245 |
| Customize toolbar in PDF Viewer component | 1246 |
| Troubleshooting | 1249 |
| Document Loading Issues in Version 23.1 or Newer | 1249 |
| Pivot Table | 1250 |
| Getting Started with ASP.NET MVC Pivot Table Control | 1250 |
| Prerequisites | 1250 |
| Create ASP.NET MVC application with HTML helper | 1250 |
| Install ASP.NET MVC package in the application | 1250 |
| Add namespace | 1251 |
| Add stylesheet and script resources | 1251 |
| Register Syncfusion script manager | 1251 |
| Add ASP.NET MVC Pivot Table control | 1252 |
| Adding fields to row, column, value and filter axes | 1255 |
| Applying formatting to a value field | 1256 |
| Enable Grouping Bar | 1257 |
| Enable Pivot Field List | 1258 |
| Calculated field | 1259 |
| Data Binding in ASP.NET MVC Pivot Table Component | 1260 |
| JSON | 1260 |
| CSV | 1263 |
| Remote Data Binding | 1266 |
| Mapping | 1270 |
| Values in row axis | 1279 |
| Values at different positions | 1280 |
| Show 'no data' items | 1281 |
| Always shows the value headers | 1282 |
| Customize empty value cells | 1283 |

| | |
|---|------|
| Events..... | 1284 |
| See Also | 1286 |
| Connecting to data source..... | 1287 |
| MySQL in EJ2 ASP.NET MVC Pivotview Component | 1287 |
| Microsoft SQL Server in EJ2 ASP.NET MVC Pivotview Component | 1291 |
| PostgreSQL in EJ2 ASP.NET MVC Pivotview Component | 1295 |
| Oracle in EJ2 ASP.NET MVC Pivotview Component | 1300 |
| MongoDB in EJ2 ASP.NET MVC Pivotview Component | 1304 |
| Elasticsearch in EJ2 ASP.NET MVC Pivotview Component..... | 1310 |
| Snowflake in EJ2 ASP.NET MVC Pivotview Component..... | 1314 |
| OLAP | 1319 |
| Getting Started with ASP.NET MVC | 1319 |
| Adding component to the application | 1325 |
| Data Binding..... | 1350 |
| OLAP Cube: Elements..... | 1355 |
| Getting Started with Syncfusion Server-side Pivot Engine | 1357 |
| Quick steps to render the Pivot Table by using the server-side Pivot Engine | 1357 |
| Available configurations in Server-side application..... | 1359 |
| Pivot Chart in ASP.NET MVC Pivot Table Component | 1376 |
| Data Binding..... | 1377 |
| Chart Types | 1377 |
| Accumulation Charts..... | 1379 |
| Field List | 1390 |
| Grouping Bar | 1391 |
| Single Axis | 1393 |
| Multiple Axis | 1395 |
| Series Customization..... | 1400 |
| Axis Customization..... | 1403 |
| Legend Customization..... | 1405 |
| User Interaction | 1407 |
| Export..... | 1411 |
| Print..... | 1413 |
| Drill Down in ASP.NET MVC Pivot Table Control | 1414 |
| Drill down and drill up..... | 1414 |
| Drill position..... | 1414 |

| | |
|---|------|
| Expand all | 1415 |
| Expand all headers for specific fields | 1416 |
| Expand all except specific member(s) | 1417 |
| Expand specific member(s) | 1418 |
| Event | 1419 |
| Data Shaping | 1424 |
| Aggregation | 1424 |
| Calculated Field in ASP.NET MVC Pivot Table Component | 1434 |
| Grouping in ASP.NET MVC Pivot Table Component | 1455 |
| Filtering | 1471 |
| Sorting in ASP.NET MVC Pivot Table Component | 1500 |
| Drill Through | 1511 |
| Editing | 1518 |
| Data Formatting | 1532 |
| Number Formatting | 1532 |
| Conditional Formatting | 1542 |
| Report Manipulation | 1553 |
| Grouping Bar | 1553 |
| Pivot Field List in ASP.NET MVC Pivot Table Component | 1579 |
| Defer Layout Update | 1615 |
| Performance | 1618 |
| Virtual Scrolling | 1618 |
| Paging in ASP.NET MVC Pivot Table Component | 1623 |
| State Persistence | 1634 |
| Save and Load Pivot Layout | 1635 |
| Row and Column in ASP.NET MVC Pivot Table Component | 1636 |
| Width and Height | 1636 |
| Row Height | 1637 |
| Column Width | 1639 |
| Reorder | 1641 |
| Column Resizing | 1642 |
| Text Wrap | 1643 |
| Text Align | 1643 |
| AutoFit | 1645 |
| Grid Lines | 1649 |

| | |
|---|------|
| Selection..... | 1650 |
| Clip Mode | 1658 |
| Cell Template | 1659 |
| Events..... | 1663 |
| See Also | 1667 |
| Show or hide totals in ASP.NET MVC Syncfusion Pivot Table Control | 1667 |
| Show or hide grand totals | 1667 |
| Show grand totals at top or bottom | 1668 |
| Show or hide sub-totals | 1669 |
| Show or hide sub-totals for specific fields | 1670 |
| Show sub-totals at top or bottom..... | 1671 |
| Show or hide totals using toolbar | 1673 |
| Hyperlink | 1675 |
| Hyperlink for all cells..... | 1675 |
| Hyperlink for row headers | 1676 |
| Hyperlink for column headers..... | 1677 |
| Hyperlink for value cells..... | 1678 |
| Hyperlink for summary cells | 1679 |
| Condition based hyperlink | 1680 |
| Condition based hyperlink for specific row or column | 1681 |
| Header based hyperlink | 1682 |
| Event | 1683 |
| Toolbar in ASP.NET MVC Syncfusion Pivot Table Control..... | 1684 |
| Built-in Toolbar Options..... | 1684 |
| Show desired chart types in the dropdown menu | 1688 |
| Switch the chart to multiple axes | 1689 |
| Show or hide legend | 1690 |
| Adding custom option to the toolbar | 1691 |
| Save and load report as a JSON file..... | 1695 |
| Save and load reports to a SQL database | 1697 |
| Events..... | 1719 |
| Tooltip | 1734 |
| Tooltip Template | 1735 |
| Style and Appearance | 1738 |
| Hiding Axis..... | 1738 |

| | |
|--|------|
| Text Alignment..... | 1739 |
| Customize header, value and summary cell style..... | 1740 |
| Printing and Exporting | 1741 |
| Print in ASP.NET MVC Pivot Table Component..... | 1741 |
| Excel Export in ASP.NET MVC Pivot Table Component..... | 1742 |
| PDF Export in ASP.NET MVC Pivot Table Component..... | 1756 |
| Globalization | 1780 |
| Internationalization..... | 1781 |
| Localization | 1787 |
| Right-to-left (RTL)..... | 1790 |
| Accessibility in Pivotview component..... | 1792 |
| WAI-ARIA attributes..... | 1793 |
| Keyboard interaction | 1794 |
| Ensuring accessibility | 1800 |
| See also | 1806 |
| Migration from Essential JS 1..... | 1806 |
| Data Binding..... | 1806 |
| Aggregation..... | 1807 |
| Number Format..... | 1808 |
| Summary Customization | 1808 |
| Drill operation | 1809 |
| Field List | 1809 |
| Grouping Bar | 1810 |
| Filtering | 1810 |
| Maximum node limit in member editor | 1810 |
| No Data Items | 1811 |
| Excel-like filtering..... | 1811 |
| Drill Through | 1811 |
| Cell Editing | 1812 |
| Hyperlink..... | 1812 |
| Defer Layout Update..... | 1813 |
| Sorting..... | 1814 |
| Value Sorting..... | 1814 |
| Calculated Field..... | 1814 |
| Paging..... | 1815 |

| | |
|---|------|
| Conditional Formatting | 1815 |
| Exporting | 1816 |
| Grid Customization | 1816 |
| Accessibility | 1817 |
| Common..... | 1817 |
| How To | 1818 |
| Switching to older themes style..... | 1818 |
| Customize number, date, and time values | 1821 |
| Customize the icons for pivot table | 1823 |
| PivotView_PivotFieldList .e-icons.e-toggle-field-list::before { | 1823 |
| Customize empty value cells..... | 1824 |
| Configure data grid options on editing mode..... | 1825 |
| Refresh the field list while change the data source..... | 1826 |
| Customizing loading indicator..... | 1827 |
| Changing the Pivot Table component's minimum height..... | 1828 |
| Render chart control based on cell selection | 1828 |
| Drill-through grid's cell edit type | 1832 |
| Show field list when pivot table is empty | 1834 |
| Apply custom style to pivot cells in ASP.NET MVC Pivot Table Component | 1835 |
| Show tooltip for row and column headers in ASP.NET MVC Pivot Table Component | 1837 |
| Hide specific columns in ASP.NET MVC Pivot Table Component | 1839 |
| Export table and chart into the same document using toolbar..... | 1840 |
| Add custom aggregation type to the menu in ASP.NET MVC Pivot Table Component | 1841 |
| Convert complex JSON to flat JSON and assign it to the pivot table | 1843 |
| Load desired report from the report list as default in ASP.NET MVC Pivot Table Component..... | 1847 |
| Display string value to pivot table values | 1849 |
| Predefined Dialogs | 1851 |
| Getting Started with ASP.NET Core Predefined Dialogs | 1851 |
| Prerequisites | 1851 |
| Create ASP.NET MVC application with HTML helper..... | 1851 |
| Install ASP.NET MVC package in the application | 1851 |
| Add namespace..... | 1851 |
| Add stylesheet and script resources | 1851 |
| Register Syncfusion script manager | 1852 |
| Open ASP.NET MVC Predefined Dialogs | 1852 |

| | |
|---|------|
| See also | 1857 |
| Draggable in Predefined Dialogs..... | 1857 |
| Animation in Predefined Dialogs | 1859 |
| Positioning in Predefined Dialogs in Blazor | 1861 |
| Dimension in Predefined Dialogs | 1864 |
| Max-width and max-height..... | 1867 |
| Min-width and min-height | 1868 |
| Customization of Predefined Dialogs..... | 1870 |
| Customize action buttons | 1870 |
| Show or hide dialog close button | 1873 |
| Customize dialog content | 1875 |
| Progress Bar | 1876 |
| Getting Started with ASP.NET MVC Progress Bar Control | 1876 |
| Prerequisites | 1876 |
| Create ASP.NET MVC application with HTML helper | 1876 |
| Install ASP.NET MVC package in the application | 1876 |
| Add namespace..... | 1877 |
| Add stylesheet and script resources | 1877 |
| Register Syncfusion script manager | 1877 |
| Add ASP.NET MVC Progress Bar control | 1877 |
| Progress Type | 1878 |
| Types | 1878 |
| Linear..... | 1878 |
| Circular | 1879 |
| Tooltip in ASP.NET MVC ProgressBar Component..... | 1879 |
| Tooltip | 1879 |
| Format..... | 1880 |
| Customization | 1880 |
| States | 1881 |
| Determinate | 1881 |
| Indeterminate | 1881 |
| Buffer | 1881 |
| Range | 1882 |
| Customization | 1882 |
| Segments..... | 1882 |

| | |
|--|------|
| Thickness..... | 1883 |
| Radius..... | 1883 |
| InnerRadius | 1883 |
| Progress color and track color | 1884 |
| Annotation and Label..... | 1884 |
| Annotation | 1884 |
| Label..... | 1885 |
| Animation..... | 1885 |
| Events..... | 1886 |
| ValueChanged | 1886 |
| ProgressCompleted..... | 1886 |
| Accessibility in ASP.NET MVC Progress bar component..... | 1887 |
| WAI-ARIA attributes..... | 1888 |
| Keyboard interaction | 1888 |
| Ensuring accessibility | 1888 |
| See also | 1888 |
| Progress Button | 1888 |
| Getting Started with ASP.NET MVC Progress Button Control | 1888 |
| Prerequisites | 1889 |
| Create ASP.NET MVC application with HTML helper..... | 1889 |
| Install ASP.NET MVC package in the application | 1889 |
| Add namespace..... | 1889 |
| Add stylesheet and script resources | 1889 |
| Register Syncfusion script manager..... | 1890 |
| Add ASP.NET MVC Progress Button control | 1890 |
| Enabling progress in button | 1890 |
| See also | 1890 |
| Spinner | 1891 |
| Progress..... | 1892 |
| See Also | 1896 |
| Accessibility in Progress Button Component | 1896 |
| WAI-ARIA attributes..... | 1897 |
| Keyboard interaction | 1897 |
| Ensuring accessibility | 1897 |
| See also | 1897 |

| | |
|--|------|
| Styles and Appearances | 1898 |
| How To | 1898 |
| Change the text content and styles of the ProgressButton during progress | 1898 |
| Customize progress using cssClass..... | 1898 |
| Hide spinner | 1899 |
| QueryBuilder | 1899 |
| Getting Started with ASP.NET MVC Query Builder Control | 1899 |
| Prerequisites | 1900 |
| Create ASP.NET MVC application with HTML helper | 1900 |
| Install ASP.NET MVC package in the application | 1900 |
| Add namespace..... | 1900 |
| Add stylesheet and script resources | 1900 |
| Register Syncfusion script manager | 1901 |
| Add ASP.NET MVC Query Builder control | 1901 |
| Render with rule | 1903 |
| Column Binding..... | 1905 |
| Auto generation | 1905 |
| Labels | 1907 |
| Operators | 1907 |
| Step | 1908 |
| Format..... | 1908 |
| Validations | 1910 |
| Template | 1912 |
| Data binding..... | 1915 |
| Local Data..... | 1915 |
| Remote data..... | 1917 |
| Support with Data Manager | 1921 |
| Grid Integration with QueryBuilder | 1923 |
| Model Binding Support | 1924 |
| Filtering | 1925 |
| Importing | 1928 |
| Initial Rendering..... | 1928 |
| Post Rendering..... | 1930 |
| Exporting..... | 1931 |
| Localization | 1933 |

| | |
|---|------|
| Styles and Appearances | 1936 |
| Header Template | 1937 |
| Accessibility in Query builder control | 1940 |
| WAI-ARIA attributes..... | 1941 |
| Keyboard interaction | 1941 |
| Ensuring accessibility | 1941 |
| See also | 1942 |
| How To | 1942 |
| Display Options | 1942 |
| Sort the columns..... | 1944 |
| Restrict the groups..... | 1946 |
| State Persistence..... | 1948 |
| Right to left (RTL) | 1950 |
| Summary View | 1951 |
| RadioButton | 1952 |
| Getting Started with ASP.NET MVC Radio Button Control | 1952 |
| Prerequisites | 1952 |
| Create ASP.NET MVC application with HTML helper..... | 1952 |
| Install ASP.NET MVC package in the application | 1952 |
| Add namespace..... | 1952 |
| Add stylesheet and script resources | 1953 |
| Register Syncfusion script manager | 1953 |
| Add ASP.NET MVC Radio Button control | 1953 |
| Change the Radio Button state..... | 1953 |
| Label and Size in Radio Button Component..... | 1954 |
| Label..... | 1954 |
| Size | 1955 |
| See Also | 1955 |
| Styles and Appearances | 1955 |
| Accessibility in Radio Button Component..... | 1956 |
| WAI-ARIA attributes..... | 1957 |
| Keyboard interaction | 1957 |
| Ensuring accessibility | 1957 |
| See also | 1957 |
| How To | 1957 |

| | |
|---|------|
| Customize RadioButton Appearance | 1957 |
| Name and Value in form submit | 1960 |
| Right-To-Left | 1960 |
| Set the disabled state..... | 1961 |
| Migration from Essential JS 1..... | 1961 |
| Properties..... | 1961 |
| Methods..... | 1963 |
| Events..... | 1963 |
| Range Navigator..... | 1963 |
| Getting Started with ASP.NET MVC Range Navigator Control..... | 1963 |
| Prerequisites | 1963 |
| Create ASP.NET MVC application with HTML helper..... | 1964 |
| Install ASP.NET MVC package in the application | 1964 |
| Add namespace..... | 1964 |
| Add script resources | 1964 |
| Register Syncfusion script manager | 1964 |
| Add ASP.NET MVC Range Navigator control..... | 1965 |
| Populate range navigator with data | 1965 |
| Selecting Range..... | 1966 |
| Lightweight range navigator | 1967 |
| See Also | 1968 |
| Series Types | 1968 |
| Line..... | 1968 |
| Area..... | 1969 |
| StepLine..... | 1970 |
| Types of data..... | 1972 |
| Numeric..... | 1972 |
| Logarithmic Axis..... | 1976 |
| Date-time | 1979 |
| Period selector | 1982 |
| Periods | 1982 |
| Positioning period selector | 1984 |
| Height..... | 1985 |
| Visibility of range navigator | 1986 |
| See Also | 1987 |

| | |
|---|------|
| Labels | 1987 |
| Multilevel labels | 1987 |
| Grouping | 1988 |
| Smart labels..... | 1989 |
| Label positioning | 1990 |
| Labels customization..... | 1991 |
| Grid and Tick Lines | 1992 |
| Grid line customization | 1992 |
| Tick line customization..... | 1993 |
| Customization | 1994 |
| Navigator appearance..... | 1994 |
| Thumb | 1995 |
| Border customization..... | 1996 |
| Deferred update..... | 1997 |
| Allow snapping..... | 1998 |
| Animation..... | 1998 |
| See Also | 1999 |
| Tooltip | 1999 |
| Enable Tooltip | 1999 |
| Customization | 2000 |
| Label Format | 2002 |
| RTL..... | 2003 |
| Export and print | 2004 |
| Export..... | 2004 |
| Print..... | 2005 |
| Accessibility in ASP.NET MVC Range navigator component..... | 2006 |
| WAI-ARIA attributes..... | 2007 |
| Keyboard interaction | 2007 |
| Ensuring accessibility | 2008 |
| See also | 2008 |
| Migration from Essential JS 1..... | 2008 |
| RangeNavigator..... | 2008 |
| Series..... | 2015 |
| StyleSettings..... | 2017 |
| Tooltip | 2019 |

| | |
|---|------|
| Period Selector | 2020 |
| Methods | 2021 |
| Events..... | 2021 |
| Range Slider | 2024 |
| Getting Started with ASP.NET MVC Range Slider Control | 2024 |
| Prerequisites | 2024 |
| Create ASP.NET MVC application with HTML helper | 2024 |
| Install ASP.NET MVC package in the application | 2024 |
| Add namespace..... | 2024 |
| Add stylesheet and script resources | 2024 |
| Register Syncfusion script manager | 2025 |
| Add ASP.NET MVC Range Slider control | 2025 |
| See also | 2025 |
| Types in RangeSlider Control | 2026 |
| Tooltip | 2027 |
| Buttons..... | 2027 |
| Orientation..... | 2028 |
| Ticks..... | 2028 |
| Step | 2029 |
| Min and Max | 2030 |
| Formatting in Range Slider Control..... | 2030 |
| Using format API | 2031 |
| Using Events | 2032 |
| Movement Limits and Drag Interval | 2032 |
| Default and MinRange Slider limits | 2033 |
| Range Slider limits..... | 2033 |
| Handle lock..... | 2034 |
| CSS structures | 2034 |
| Customizing the slider track..... | 2034 |
| Customizing the slider handle..... | 2035 |
| Customizing the slider limits | 2035 |
| Customizing the slider ticks | 2035 |
| Customizing the slider buttons | 2035 |
| Accessibility in ASP.NET MVC Range Slider component | 2036 |
| WAI-ARIA attributes..... | 2037 |

| | |
|---|------|
| Keyboard interaction | 2037 |
| Ensuring accessibility | 2037 |
| See also | 2038 |
| Migration from Essential JS 1..... | 2038 |
| How To | 2039 |
| Time Range Slider | 2039 |
| Date Range Slider | 2040 |
| Numeric Range Slider..... | 2041 |
| Customize Slider Bar | 2042 |
| Customize Slider Limits | 2046 |
| Customize Slider Ticks label | 2048 |
| ticks_slider .e-scale :nth-child(1)::before { | 2048 |
| Customize Slider Thumb | 2052 |
| square_slider.e-control.e-slider .e-handle { | 2052 |
| circle_slider.e-control.e-slider .e-handle {..... | 2052 |
| oval_slider.e-control.e-slider .e-handle {..... | 2052 |
| Form Slider with FormValidator..... | 2055 |
| Show Slider from Hidden State | 2060 |
| Reversible Range Slider..... | 2061 |
| Rating | 2062 |
| Getting Started with ASP.NET MVC Rating Control | 2062 |
| Prerequisites | 2062 |
| Create ASP.NET MVC application with HTML helper..... | 2062 |
| Install ASP.NET MVC package in the application | 2063 |
| Add namespace..... | 2063 |
| Add stylesheet and script resources | 2063 |
| Register Syncfusion script manager | 2063 |
| Add ASP.NET MVC Rating control | 2064 |
| Value | 2064 |
| Precision Modes in ASP.NET MVC Rating Control | 2064 |
| Labels in ASP.NET MVC Rating Control | 2065 |
| Label position..... | 2066 |
| Label template | 2067 |
| Tooltip in ASP.NET MVC Rating Control..... | 2067 |
| Tooltip template | 2068 |

| | |
|---|------|
| Tooltip customization | 2068 |
| Selection in ASP.NET MVC Rating Control | 2069 |
| Min value | 2070 |
| Single selection | 2070 |
| Show or hide reset button | 2071 |
| Templates in ASP.NET MVC Rating Control | 2071 |
| Empty (unrated) symbol template | 2071 |
| Full (rated) symbol template | 2073 |
| Using Emoji icon as rating symbol | 2074 |
| Using SVG icon as rating symbol | 2075 |
| Using PNG image as rating symbol | 2076 |
| Events in Rating Control | 2077 |
| BeforeItemRender | 2077 |
| Created | 2077 |
| ValueChanged | 2078 |
| OnItemHover | 2078 |
| Appearance in ASP.NET MVC Rating Control | 2078 |
| Items count | 2079 |
| Disabled | 2079 |
| Visible | 2079 |
| Read only | 2080 |
| CssClass | 2080 |
| Changing icon using CssClass | 2082 |
| Accessibility in ASP.NET MVC Rating Control | 2083 |
| Keyboard interaction | 2083 |
| ARIA attribute | 2083 |
| Ribbon | 2084 |
| Getting Started with ASP.NET MVC Ribbon Control | 2084 |
| Prerequisites | 2084 |
| Create ASP.NET MVC application with HTML helper | 2084 |
| Install ASP.NET MVC package in the application | 2084 |
| Add namespace | 2084 |
| Add stylesheet and script resources | 2085 |
| Register Syncfusion script manager | 2085 |
| Add ASP.NET MVC Ribbon control | 2085 |

| | |
|-------------------------------|------|
| Adding Ribbon Tab | 2085 |
| Adding Ribbon Group..... | 2086 |
| Adding Ribbon Items..... | 2086 |
| Tabs and Groups | 2091 |
| Adding Tabs..... | 2091 |
| Adding Groups | 2092 |
| Adding Items | 2092 |
| Ribbon Items | 2093 |
| Built-in items..... | 2093 |
| Custom items | 2105 |
| Items display Mode..... | 2106 |
| Enable or disable items | 2107 |
| Ribbon Layouts..... | 2108 |
| Classic layout..... | 2108 |
| Simplified layout | 2122 |
| Minimized state | 2126 |
| File Menu | 2127 |
| Visibility..... | 2127 |
| Adding Menu Items..... | 2128 |
| Open Submenu on click | 2129 |
| Custom Header text | 2131 |
| Ribbon Backstage..... | 2132 |
| Adding backstage items | 2132 |
| Adding footer items | 2135 |
| Adding separator..... | 2138 |
| Back button..... | 2142 |
| Backstage target..... | 2144 |
| Template | 2147 |
| Setting width and height..... | 2152 |
| Help pane | 2155 |
| Tooltip | 2156 |
| Adding Title | 2156 |
| Adding Content | 2157 |
| Adding Icon | 2158 |
| Customization | 2159 |

| | |
|---|------|
| Ribbon Resizing | 2161 |
| Defining items allowed size | 2161 |
| Defining items active size..... | 2161 |
| Events..... | 2161 |
| Tab selected | 2162 |
| Tab selecting | 2162 |
| Ribbon collapsing | 2163 |
| Ribbon expanding | 2164 |
| Group launcher click | 2164 |
| Button item events | 2165 |
| Checkbox item events | 2166 |
| Colorpicker item events | 2167 |
| ComboBox item events | 2171 |
| DropDown item events | 2176 |
| SplitButton item events | 2180 |
| GroupButton item events | 2186 |
| FileMenu events..... | 2187 |
| Backstage Menu events | 2192 |
| Methods..... | 2193 |
| addTab | 2193 |
| addGroup | 2194 |
| addCollection | 2195 |
| addItem | 2196 |
| removeTab | 2197 |
| removeGroup..... | 2197 |
| removeCollection..... | 2198 |
| removeItem..... | 2199 |
| enableItem | 2199 |
| disableItem..... | 2200 |
| refreshLayout | 2200 |
| selectTab | 2201 |
| RichTextEditor | 2202 |
| Getting Started with ASP.NET MVC Rich Text Editor Control | 2202 |
| Prerequisites | 2202 |
| Create ASP.NET MVC application with HTML helper | 2202 |

| | |
|--|------|
| Install ASP.NET MVC package in the application | 2202 |
| Add namespace..... | 2202 |
| Add stylesheet and script resources | 2202 |
| Register Syncfusion script manager | 2203 |
| Add ASP.NET MVC Rich Text Editor control | 2203 |
| Configure the Toolbar | 2206 |
| Insert Images and Links..... | 2208 |
| Send formatted content using XmlHttpRequest..... | 2209 |
| Retrieve the Formatted Content..... | 2211 |
| Retrieve the number of characters..... | 2211 |
| See also | 2212 |
| Formation..... | 2212 |
| HTML Editor | 2212 |
| Markdown Editor | 2213 |
| See Also | 2214 |
| Toolbar | 2214 |
| Expand Toolbar | 2215 |
| Multi-row Toolbar | 2215 |
| Floating Toolbar | 2216 |
| Toolbar Items | 2217 |
| Custom Tool | 2222 |
| Quick Inline Toolbar | 2226 |
| See Also | 2228 |
| Styling..... | 2228 |
| Font Name and Font Size | 2228 |
| Custom Fonts and Size | 2229 |
| Font and Background Color | 2231 |
| Editor content styles | 2232 |
| See Also | 2238 |
| Image | 2238 |
| Upload Options | 2239 |
| Server Side Action | 2240 |
| Image Delete | 2241 |
| Insert from Web..... | 2243 |
| Dimension | 2243 |

| | |
|---|------|
| Caption and Alt Text..... | 2243 |
| Display Position..... | 2243 |
| Image with Link..... | 2244 |
| Drag and Drop..... | 2245 |
| See Also..... | 2246 |
| Insert Audio..... | 2246 |
| Configure audio tool in the toolbar | 2247 |
| Insert audio from web | 2247 |
| Insert from web URL | 2247 |
| Upload and insert audio..... | 2248 |
| Replacing audio..... | 2250 |
| Delete audio | 2251 |
| Display Position..... | 2253 |
| Rename audio before inserting | 2254 |
| Upload audio with authentication | 2256 |
| See Also | 2257 |
| Insert Video..... | 2257 |
| Configure video tool in the toolbar..... | 2258 |
| Insert video from web..... | 2258 |
| Insert from web URL | 2258 |
| Upload and insert video | 2259 |
| Replacing video | 2262 |
| Delete video | 2263 |
| Dimension | 2264 |
| Display Position..... | 2265 |
| Resize video..... | 2266 |
| Rename video before inserting..... | 2266 |
| Upload video with authentication | 2268 |
| See Also | 2269 |
| Link..... | 2269 |
| Insert Link..... | 2269 |
| Remove Link..... | 2270 |
| Auto-link..... | 2271 |
| Manipulation..... | 2271 |
| See Also | 2272 |

| | |
|--|------|
| Table..... | 2272 |
| Insert Table | 2272 |
| Quick Toolbar | 2272 |
| Table Header..... | 2273 |
| Insert Rows..... | 2273 |
| Insert Columns | 2273 |
| Set Color..... | 2273 |
| Delete Table | 2274 |
| Vertical Align | 2274 |
| Horizontal Align..... | 2274 |
| Table Styles | 2274 |
| Table Properties | 2275 |
| Table cell merge and split | 2275 |
| Emoji Picker..... | 2276 |
| Enabling the toolbar option and custom emojis..... | 2276 |
| Using the shortcut key to open the emoji picker..... | 2279 |
| Navigating and selecting emojis using the keyboard..... | 2279 |
| Markdown..... | 2280 |
| Supported Commands | 2280 |
| Markdown to HTML | 2281 |
| Table..... | 2284 |
| See Also | 2289 |
| File Browser | 2289 |
| Format Painter in ASP.NET MVC Rich Text Editor Control Syncfusion..... | 2290 |
| Enabling the toolbar option for Format Painter | 2291 |
| Using the shortcut key to copy and paste the format | 2294 |
| Form support | 2294 |
| Render the Rich Text Editor | 2294 |
| Obtain the Value | 2294 |
| See Also | 2296 |
| Validation | 2296 |
| Validation Rules | 2296 |
| Validation Message..... | 2298 |
| Custom Placement of Validation Message | 2299 |
| Globalization | 2300 |

| | |
|---|------|
| Localization | 2300 |
| RTL..... | 2313 |
| XHTML validation | 2314 |
| Attributes | 2314 |
| HTML Elements | 2314 |
| Miscellaneous | 2315 |
| Placeholder | 2315 |
| Character Count | 2316 |
| Print..... | 2317 |
| Code View | 2318 |
| Full Screen | 2321 |
| Prevention of cross-site scripting (XSS) | 2322 |
| Resizable support..... | 2323 |
| Number and Bullet Format Lists | 2324 |
| Third party Integration..... | 2325 |
| Code-Mirror Integration | 2325 |
| Embed.ly Integration..... | 2329 |
| Inline Mode..... | 2330 |
| Show on Select/Click..... | 2330 |
| See Also | 2331 |
| Paste from MS Word..... | 2332 |
| MS Word to HTML | 2332 |
| Paste cleanup | 2332 |
| Prompt dialog..... | 2332 |
| Paste as plain text | 2333 |
| Keep format | 2333 |
| Denied tags | 2333 |
| Denied attributes | 2333 |
| Allowed style properties | 2333 |
| Mention Integration with Rich Text Editor | 2334 |
| See Also | 2338 |
| Enter and Shift-Enter Key's Customization | 2338 |
| Enter key customization..... | 2338 |
| Shift-Enter key customization | 2339 |
| Iframe..... | 2339 |

| | |
|--|------|
| Iframe Attributes..... | 2340 |
| Adding External CSS/Script File..... | 2341 |
| See Also | 2342 |
| ExecCommand in Rich Text Editor | 2342 |
| CSS structures | 2344 |
| Customizing the Rich Text Editor's content | 2344 |
| Customizing the Rich Text Editor's toolbar | 2344 |
| Customizing the Rich Text Editor's character count | 2345 |
| Keyboard Support | 2345 |
| HTML Formation Shortcut Key..... | 2345 |
| Markdown Formation Shortcut Key..... | 2348 |
| Custom Key Config | 2349 |
| See Also | 2350 |
| Accessibility..... | 2350 |
| ARIA Attributes | 2351 |
| Keyboard interaction | 2353 |
| Ensuring accessibility | 2355 |
| See also | 2355 |
| How To | 2355 |
| Add Google fonts | 2355 |
| Customize the placeholder style..... | 2357 |
| Restrict the image uploading while uploading with large size | 2357 |
| Save Rich Text Editor content in a file in the server | 2358 |
| Set the cursor at the specific range | 2359 |
| Capture ctrl+s to update the value | 2360 |
| Render the RichTextEditorFor control..... | 2361 |
| Rename uploaded images in server before inserting it in the Rich Text Editor | 2363 |
| File Attachments | 2364 |
| Format code block using toolbar button | 2367 |
| Migration from Essential JS 1..... | 2369 |
| Accessibility..... | 2369 |
| Toolbar | 2370 |
| Custom Formats and Fonts | 2371 |
| Custom Font Colors..... | 2372 |
| Link | 2372 |

| | |
|--|------|
| Image..... | 2372 |
| Table..... | 2373 |
| Counts | 2374 |
| IFrame | 2374 |
| Editor Mode | 2375 |
| Undo..... | 2375 |
| Common..... | 2375 |
| Execute Command | 2380 |
| Schedule | 2380 |
| Getting Started with ASP.NET MVC Scheduler Control | 2380 |
| Prerequisites | 2380 |
| Create ASP.NET MVC application with HTML helper..... | 2380 |
| Install ASP.NET MVC package in the application | 2381 |
| Add namespace..... | 2381 |
| Add stylesheet and script resources | 2381 |
| Register Syncfusion script manager | 2381 |
| Add ASP.NET MVC Scheduler control | 2382 |
| Populating appointments | 2382 |
| Setting date | 2384 |
| Setting view..... | 2385 |
| Individual view customization | 2386 |
| ASP.NET MVC Scaffolding | 2388 |
| Getting Started..... | 2388 |
| Scheduler interactions | 2393 |
| Appointments in ASP.NET MVC Schedule Component..... | 2394 |
| Normal events..... | 2395 |
| Spanned events..... | 2395 |
| All-day events..... | 2396 |
| Customize the rendering of the spanned events..... | 2396 |
| Recurring events | 2397 |
| Event fields..... | 2404 |
| Adding Custom fields | 2408 |
| Customize the order of the overlapping events | 2409 |
| Drag and drop appointments..... | 2410 |
| Inline Appointment | 2421 |

| | |
|---|------|
| Appointment Resizing | 2422 |
| Appointment customization | 2426 |
| Setting minimum height | 2432 |
| Block Dates and Times | 2433 |
| Readonly | 2434 |
| Make specific events readonly..... | 2435 |
| Restricting event creation on specific time slots | 2436 |
| Differentiate the past time events..... | 2438 |
| Appointments occupying entire cell | 2439 |
| How to limit maximum number of events to display | 2440 |
| Display tooltip for appointments | 2442 |
| Appointment selection | 2445 |
| Deleting multiple appointments | 2445 |
| Retrieve event details from the UI of an event | 2445 |
| Get the current view appointments | 2447 |
| Get the entire appointment collections..... | 2450 |
| Refresh appointments | 2452 |
| Data-binding..... | 2452 |
| Binding local data..... | 2452 |
| Binding remote data | 2453 |
| Loading data via AJAX post | 2456 |
| Passing additional parameters to the server | 2457 |
| Handling failure actions | 2458 |
| Scheduler CRUD actions..... | 2458 |
| Configuring Scheduler with Google API service..... | 2462 |
| CRUD Actions | 2463 |
| Add | 2463 |
| Edit | 2470 |
| Delete..... | 2482 |
| Drag and drop | 2490 |
| Resize | 2490 |
| Virtual scrolling in ASP.NET MVC Schedule control | 2490 |
| Enabling lazy loading for appointments..... | 2492 |
| Editor window and quick popups | 2495 |
| Event editor..... | 2495 |

| | |
|--|------|
| Quick popups | 2503 |
| Customizing event editor using template | 2509 |
| More events indicator and popup | 2524 |
| Timezone..... | 2534 |
| Understanding date manipulation in JavaScript..... | 2534 |
| Scheduler with no timezone | 2534 |
| Scheduler set to specific timezone | 2535 |
| Display events on same time everywhere with no time difference | 2536 |
| Set specific timezone for events | 2537 |
| Add or remove timezone names to/from the timezone collection | 2538 |
| Timezone methods | 2539 |
| Views | 2541 |
| Setting specific view on scheduler | 2541 |
| View specific configuration | 2543 |
| Extending view intervals | 2559 |
| See Also | 2560 |
| Calendar mode..... | 2560 |
| Gregorian Calendar | 2560 |
| Islamic Calendar | 2560 |
| Multiple resources and grouping | 2562 |
| Resource fields | 2563 |
| Resource data binding | 2564 |
| Scheduler with multiple resources | 2566 |
| Resource grouping | 2567 |
| Customizing parent resource cells | 2579 |
| Working with shared events | 2582 |
| Simple resource header customization | 2583 |
| Customizing resource header with multiple columns | 2586 |
| Collapse/Expand child resources in timeline views | 2590 |
| Displaying tooltip for resource headers..... | 2593 |
| Choosing between resource colors for appointments | 2594 |
| Dynamically add and remove resources | 2597 |
| Setting different working days and hours for resources | 2599 |
| Hide non-working days when grouped by date | 2603 |
| Compact view in mobile..... | 2605 |

| | |
|---|------|
| Adaptive UI in desktop..... | 2606 |
| Timeline header rows | 2609 |
| Display year and month rows in timeline views | 2610 |
| Display week numbers in timeline views | 2610 |
| Timeline view displaying dates of a complete year | 2611 |
| Customizing the header rows using template | 2611 |
| Row Auto Height | 2612 |
| Calendar month view | 2613 |
| Timeline views..... | 2614 |
| Timeline views with multiple resources | 2616 |
| Appointments occupying entire cell | 2618 |
| Header customization | 2620 |
| Show or Hide header bar | 2620 |
| Customizing header bar using template | 2621 |
| Customizing header bar using events | 2627 |
| How to display the view options within the header bar popup | 2630 |
| Date header customization..... | 2631 |
| Customizing the date range text..... | 2634 |
| TimeScale Customization | 2636 |
| Setting different time slot duration | 2636 |
| Customizing time cells using template | 2637 |
| Hide the timescale | 2638 |
| Highlighting current date and time..... | 2639 |
| Setting working days and hours..... | 2640 |
| Set working days | 2640 |
| Hiding weekend days | 2641 |
| Show week numbers..... | 2642 |
| Set working hours | 2645 |
| Scheduler displaying custom hours | 2646 |
| Setting start day of the week | 2647 |
| Scroll to specific time and date | 2648 |
| See Also | 2650 |
| Cell Customization | 2650 |
| Setting cell dimensions in all views..... | 2650 |
| Check for cell availability..... | 2651 |

| | |
|--|------|
| Customizing cells in all the views | 2653 |
| Customizing cell header in month view | 2656 |
| Customizing the minimum and maximum date values | 2656 |
| How to disable multiple cell and row selection in Schedule | 2657 |
| Set state persistence..... | 2657 |
| Exporting in ASP.NET MVC Schedule Component | 2659 |
| Excel Exporting | 2659 |
| Exporting calendar events as ICS file | 2671 |
| Import events from other calendars..... | 2675 |
| How to print the Scheduler element | 2677 |
| Context menu | 2681 |
| Scheduler dimensions | 2685 |
| Auto Height and Width | 2685 |
| Height and Width in pixel | 2686 |
| Height and Width in percentage..... | 2687 |
| See Also | 2688 |
| Recurrence editor | 2688 |
| Customizing the repeat type option in editor | 2688 |
| Customizing the End Type Option in Editor | 2689 |
| Accessing the recurrence rule string..... | 2690 |
| Set specific value on recurrence editor | 2691 |
| Recurrence date generation | 2692 |
| Recurrence date generation in server-side..... | 2694 |
| Restrict date generation with specific count | 2694 |
| Globalization and Localization | 2695 |
| Globalization | 2695 |
| Localizing the static Scheduler text..... | 2698 |
| Setting date format..... | 2701 |
| Setting the time format | 2702 |
| Displaying Scheduler in RTL mode | 2703 |
| See Also | 2704 |
| Accessibility in ASP.NET MVC Schedule control..... | 2704 |
| ARIA attributes..... | 2705 |
| Keyboard interaction | 2706 |
| Ensuring accessibility | 2706 |

| | |
|--|------|
| See also | 2707 |
| Styling and appearance..... | 2707 |
| Migration from Essential JS 1..... | 2708 |
| Scheduler | 2708 |
| Properties..... | 2708 |
| Methods..... | 2717 |
| Events..... | 2722 |
| How To | 2726 |
| Perform CRUD Actions Dynamically..... | 2726 |
| Set Default Value for Event Fields..... | 2730 |
| Open Editor Window in different ways | 2731 |
| Prevent the Date Navigation..... | 2734 |
| Half-yearly view | 2735 |
| Set Different Working Hours on Different Days | 2737 |
| Show quick info Template..... | 2738 |
| Enable scroll option on all-day section | 2746 |
| Refresh Layout | 2747 |
| Set Different Time Duration on Event Editor | 2748 |
| Prioritize the Resource Color for Events | 2749 |
| SideBar | 2750 |
| Getting Started with ASP.NET MVC Sidebar Control | 2750 |
| Prerequisites | 2750 |
| Create ASP.NET MVC application with HTML helper..... | 2751 |
| Install ASP.NET MVC package in the application | 2751 |
| Add namespace..... | 2751 |
| Add stylesheet and script resources | 2751 |
| Register Syncfusion script manager..... | 2752 |
| Add ASP.NET MVC Sidebar control | 2752 |
| Enable backdrop | 2753 |
| Position | 2755 |
| Animate..... | 2760 |
| Close on document click | 2761 |
| Enable gestures..... | 2763 |
| See also | 2765 |
| Context..... | 2765 |

| | |
|--|------|
| See Also | 2769 |
| Types | 2769 |
| Expanding types of Sidebar | 2769 |
| See Also | 2775 |
| Auto-close | 2775 |
| Dock | 2778 |
| See Also | 2784 |
| Styles and Appearance..... | 2784 |
| Customizing the sidebar..... | 2784 |
| Customizing the sidebar based on the positions | 2784 |
| Customizing the sidebar based on the active state | 2785 |
| Customizing the sidebar with dock state..... | 2785 |
| Customizing the different types of sidebar..... | 2786 |
| Customizing the backdrop of the sidebar | 2786 |
| Customizing the sidebar in the RTL direction | 2787 |
| Prevent the animation transition for the Sidebar component | 2787 |
| Accessibility in ASP.NET MVC Sidebar component..... | 2787 |
| WAI-ARIA attributes..... | 2788 |
| Keyboard interaction | 2788 |
| Ensuring accessibility | 2788 |
| See also | 2789 |
| How To | 2789 |
| Initialize the Sidebar with ListView | 2789 |
| Open and close the Sidebar | 2792 |
| Multiple Sidebar in SideBar Control..... | 2795 |
| Sidebar with fixed position | 2797 |
| Custom animation effects with sidebar | 2811 |
| Layout Sidebar | 2815 |
| Initialize the Sidebar with TreeView | 2819 |
| Layout Sidebar using Content Template..... | 2829 |
| Hide Sidebar | 2837 |
| Sidebar with partial view | 2842 |
| Adding Dropdownlist inside the sidebar..... | 2845 |
| Signature | 2847 |
| Getting Started with ASP.NET MVC Signature Control | 2847 |

| | |
|---|------|
| Prerequisites | 2847 |
| Create ASP.NET MVC application with HTML helper | 2847 |
| Install ASP.NET MVC package in the application | 2847 |
| Add namespace..... | 2848 |
| Add stylesheet and script resources | 2848 |
| Register Syncfusion script manager | 2848 |
| Add ASP.NET MVC Signature control..... | 2849 |
| Customization of Signature Control..... | 2849 |
| Stroke Width | 2849 |
| Stroke Color | 2850 |
| Background Color | 2852 |
| Background Image | 2853 |
| See Also | 2855 |
| Open and Save Signature..... | 2855 |
| Open Signature | 2855 |
| Save Signature..... | 2857 |
| Save with Background..... | 2861 |
| Draw a Signature..... | 2865 |
| Draw | 2865 |
| User Interactions..... | 2867 |
| Undo and Redo | 2867 |
| Disabled..... | 2868 |
| ReadOnly | 2868 |
| Accessibility in Signature Component | 2870 |
| Keyboard interaction | 2871 |
| Ensuring accessibility | 2872 |
| See also | 2872 |
| How To | 2872 |
| Integration with Toolbar | 2872 |
| Skeleton | 2880 |
| Getting Started with ASP.NET MVC Skeleton Control | 2880 |
| Prerequisites | 2880 |
| Create ASP.NET MVC application with HTML helper | 2880 |
| Install ASP.NET MVC package in the application | 2881 |
| Add namespace..... | 2881 |

| | |
|---|------|
| Add stylesheet and script resources | 2881 |
| Register Syncfusion script manager | 2881 |
| Add ASP.NET MVC Skeleton control | 2882 |
| Skeleton Types | 2882 |
| Shapes in ASP.NET MVC Skeleton Control | 2884 |
| Circle skeleton shape | 2884 |
| Square skeleton shape | 2884 |
| Rectangle skeleton shape | 2885 |
| Text skeleton shape | 2885 |
| Shimmer Effect in ASP.NET MVC Skeleton Control | 2887 |
| Styles in ASP.NET MVC Skeleton Control | 2889 |
| CssClass | 2889 |
| Visible | 2889 |
| Accessibility in ASP.NET MVC Skeleton Control | 2890 |
| ARIA attributes | 2890 |
| Smith Chart | 2890 |
| Getting Started with ASP.NET MVC SmithChart Control | 2890 |
| Prerequisites | 2890 |
| Create ASP.NET MVC application with HTML helper | 2890 |
| Install ASP.NET MVC package in the application | 2890 |
| Add namespace | 2891 |
| Add script resources | 2891 |
| Register Syncfusion script manager | 2891 |
| Add ASP.NET MVC SmithChart control | 2891 |
| Working with Data | 2892 |
| Data Binding | 2892 |
| Smithchart Dimensions | 2894 |
| Size for Container | 2894 |
| Size for Smithchart | 2895 |
| Title and Subtitle | 2897 |
| Enable title | 2897 |
| Title trim | 2898 |
| Axis in Smithchart Control | 2899 |
| Labels Customization | 2899 |
| Gridlines | 2901 |

| | |
|--|------|
| Axisline | 2903 |
| Legend | 2904 |
| Position and Alignment | 2904 |
| Customization | 2908 |
| Toggle Visibility | 2912 |
| Tooltip | 2914 |
| Marker & Datalabels | 2915 |
| Marker | 2915 |
| Datalabels | 2918 |
| Series | 2920 |
| points or datasource | 2920 |
| Series customization | 2921 |
| Print and Export | 2923 |
| Print | 2923 |
| Export | 2924 |
| Accessibility in ASP.NET MVC Smith chart component | 2926 |
| WAI-ARIA attributes | 2927 |
| Keyboard interaction | 2927 |
| Ensuring accessibility | 2927 |
| See also | 2927 |
| Sparkline | 2928 |
| Getting Started with ASP.NET MVC Sparkline Control | 2928 |
| Prerequisites | 2928 |
| Create ASP.NET MVC application with HTML helper | 2928 |
| Install ASP.NET MVC package in the application | 2928 |
| Add namespace | 2928 |
| Add script resources | 2928 |
| Register Syncfusion script manager | 2929 |
| Add ASP.NET MVC Sparkline control | 2929 |
| Sparkline Dimensions | 2930 |
| Size for container | 2930 |
| Size for sparkline | 2931 |
| Sparkline Types | 2934 |
| Axis Customization | 2942 |
| Change value type of the sparkline | 2942 |

| | |
|--|------|
| Change min and max values of axis | 2946 |
| Change value of axis..... | 2948 |
| Axis line customization | 2949 |
| Special points customization | 2950 |
| Range Band | 2953 |
| Range band customization..... | 2953 |
| Multiple range band customization | 2954 |
| Markers | 2955 |
| Adding marker to the sparkline | 2955 |
| Adding marker to special point..... | 2956 |
| Customizing markers..... | 2957 |
| Data Labels..... | 2958 |
| Enable data label..... | 2958 |
| Customize data label..... | 2960 |
| Format data label text..... | 2961 |
| User interactions..... | 2962 |
| Tooltip | 2962 |
| Track line..... | 2966 |
| Appearance | 2967 |
| Sparkline border..... | 2967 |
| Sparkline padding..... | 2968 |
| Sparkline area customization..... | 2969 |
| Sparkline theme | 2970 |
| Localization | 2971 |
| Tooltip format | 2971 |
| Accessibility in ASP.NET MVC Sparkline component | 2972 |
| WAI-ARIA attributes..... | 2973 |
| Keyboard interaction | 2973 |
| Ensuring accessibility | 2974 |
| See also | 2974 |
| Migration from Essential JS 1..... | 2974 |
| Sparkline Types | 2974 |
| Databinding..... | 2974 |
| Markers | 2974 |
| Data labels..... | 2975 |

| | |
|--|------|
| Range band | 2976 |
| Special points customization | 2977 |
| Axis customization | 2977 |
| Appearance customization | 2978 |
| Tooltip | 2979 |
| Rendering..... | 2981 |
| Localization | 2981 |
| Methods..... | 2981 |
| Events..... | 2981 |
| SpeedDial | 2983 |
| Getting Started with ASP.NET MVC SpeedDial Control | 2983 |
| Prerequisites | 2983 |
| Create ASP.NET MVC application with HTML helper..... | 2983 |
| Install ASP.NET MVC package in the application | 2983 |
| Add namespace..... | 2983 |
| Add stylesheet and script resources | 2983 |
| Register Syncfusion script manager | 2984 |
| Add ASP.NET MVC SpeedDial control | 2984 |
| Positioning..... | 2985 |
| Linear and radial display modes | 2986 |
| Clicked event..... | 2988 |
| Items in ASP.NET MVC Speed Dial Control | 2989 |
| Icons in speeddial items..... | 2989 |
| Animation..... | 2992 |
| Template | 2993 |
| Positions in ASP.NET MVC Speed Dial Control..... | 2993 |
| Opens on hover..... | 2994 |
| Programmatically show/hide | 2995 |
| Programmatically refresh the position | 2997 |
| Display Modes in ASP.NET MVC Speed Dial Control..... | 2998 |
| Linear display mode | 2998 |
| Radial display mode (Radial Menu) | 2999 |
| Radial Menu in ASP.NET MVC Speed Dial Control..... | 2999 |
| Radial Menu direction..... | 2999 |
| Start and end angle | 3000 |

| | |
|--|------|
| Offset..... | 3001 |
| Template in ASP.NET MVC SpeedDial Control | 3001 |
| Item template | 3002 |
| Popup template | 3003 |
| Styles in ASP.NET MVC SpeedDial Control..... | 3003 |
| SpeedDial button | 3004 |
| Disabled..... | 3006 |
| CssClass | 3007 |
| Visible..... | 3008 |
| Tooltip | 3008 |
| Opens on hover..... | 3009 |
| Modal in ASP.NET MVC SpeedDial Control | 3010 |
| Events in ASP.NET MVC SpeedDial Control..... | 3011 |
| Clicked event..... | 3011 |
| Created..... | 3011 |
| Before open | 3012 |
| On open..... | 3013 |
| Before close | 3013 |
| On close..... | 3014 |
| Before item render | 3014 |
| Accessibility in ASP.NET MVC SpeedDial Control..... | 3015 |
| Keyboard interaction | 3015 |
| ARIA attributes..... | 3016 |
| Spinner | 3016 |
| Getting Started with ASP.NET MVC Spinner Control | 3016 |
| Prerequisites | 3016 |
| Create ASP.NET MVC application with HTML helper..... | 3016 |
| Install ASP.NET MVC package in the application | 3016 |
| Add namespace..... | 3016 |
| Add stylesheet and script resources | 3017 |
| Register Syncfusion script manager | 3017 |
| Add ASP.NET MVC Spinner Control | 3017 |
| Set the template to the Spinner | 3018 |
| Change the type of the Spinner | 3023 |
| CSS structures | 3025 |

| | |
|---|------|
| Customizing the spinner | 3025 |
| How To | 3026 |
| Real Time Example Using Spinner | 3026 |
| Split Button | 3031 |
| Getting Started with ASP.NET MVC Split Button Control | 3031 |
| Prerequisites | 3031 |
| Create ASP.NET MVC application with HTML helper | 3031 |
| Install ASP.NET MVC package in the application | 3031 |
| Add namespace..... | 3031 |
| Add stylesheet and script resources | 3032 |
| Register Syncfusion script manager | 3032 |
| Add ASP.NET MVC Split Button control | 3032 |
| See also | 3033 |
| Icons in SplitButton Control | 3033 |
| SplitButton Icons | 3033 |
| Vertical Button | 3035 |
| See Also | 3037 |
| Popup Items | 3037 |
| Icons | 3037 |
| Template | 3039 |
| See Also | 3046 |
| Accessibility in Split Button Controls | 3046 |
| WAI-ARIA attributes..... | 3047 |
| Keyboard interaction | 3047 |
| Ensuring accessibility | 3047 |
| See also | 3048 |
| Styles and Appearances | 3048 |
| How To | 3048 |
| Create right-to-left SplitButton..... | 3048 |
| Group items in Popup | 3050 |
| Open a dialog on popup item click | 3051 |
| Set the disabled state..... | 3052 |
| Underline a character in a text | 3054 |
| Migration from Essential JS 1..... | 3055 |
| Properties..... | 3055 |

| | |
|---|------|
| Methods..... | 3057 |
| Events..... | 3057 |
| Splitter..... | 3059 |
| Getting Started with ASP.NET MVC Splitter Control..... | 3059 |
| Prerequisites | 3059 |
| Create ASP.NET MVC application with HTML helper | 3059 |
| Install ASP.NET MVC package in the application | 3059 |
| Add namespace..... | 3059 |
| Add stylesheet and script resources | 3060 |
| Register Syncfusion script manager | 3060 |
| Add ASP.NET MVC Splitter control..... | 3060 |
| Load content to the pane..... | 3061 |
| Configure pane size..... | 3062 |
| Resizable panes..... | 3063 |
| Set minimum and maximum pane sizes | 3064 |
| Orientation..... | 3064 |
| Nested Splitter | 3065 |
| See also | 3067 |
| Split panes..... | 3067 |
| Horizontal layout..... | 3067 |
| Vertical layout..... | 3068 |
| Multiple panes | 3069 |
| Separator..... | 3071 |
| Nested Splitter | 3072 |
| Add or remove pane | 3074 |
| See Also | 3076 |
| Expand and Collapse | 3077 |
| Collapsible panes | 3077 |
| Programmatically control the expand and collapse action | 3078 |
| Specify initial state to panes | 3079 |
| See Also | 3080 |
| Resize | 3080 |
| Min and Max validation | 3080 |
| Prevent resizing..... | 3081 |
| Refresh content on resizing | 3082 |

| | |
|--|------|
| Customize the resize grip and cursor | 3082 |
| See Also | 3083 |
| Pane content in Splitter Control | 3083 |
| HTML Markup | 3083 |
| Syncfusion ASP.NET MVC UI controls | 3084 |
| Plain content | 3084 |
| Pane content using selector | 3085 |
| Pane sizing..... | 3087 |
| Auto size panes | 3088 |
| Fixed pane | 3089 |
| Different layouts | 3090 |
| Code editor style layout..... | 3090 |
| code-text { | 3092 |
| target { | 3092 |
| Outlook style layout | 3093 |
| discard { | 3096 |
| target { | 3096 |
| groupedList.e-listview .e-list-group-item { | 3097 |
| splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar { | 3097 |
| buttonSection { | 3097 |
| createpostholder { | 3097 |
| See Also | 3097 |
| CSS structures | 3097 |
| Customizing the split bar | 3097 |
| Customizing the split bar resize handle | 3098 |
| Customizing the split bar arrows | 3099 |
| To hide the resize handle in Splitter | 3100 |
| Expand and Collapse | 3100 |
| Collapsible panes | 3100 |
| See Also | 3101 |
| Keyboard interaction | 3102 |
| Ensuring accessibility | 3102 |
| See also | 3103 |
| Migration from Essential JS 1..... | 3103 |
| Common..... | 3103 |

| | |
|--|------|
| Accessibility and Localization | 3104 |
| Control State | 3104 |
| State Maintenance | 3104 |
| Pane Properties | 3104 |
| Animation | 3107 |
| SpreadSheet | 3107 |
| Overview of the ASP.NET MVC Spreadsheet control | 3107 |
| Key features | 3107 |
| Getting Started with ASP.NET MVC Spreadsheet Control | 3108 |
| Prerequisites | 3108 |
| Create ASP.NET MVC application with HTML helper | 3109 |
| Install ASP.NET MVC package in the application | 3109 |
| Add namespace | 3109 |
| Add stylesheet and script resources | 3109 |
| Register Syncfusion script manager | 3110 |
| Add ASP.NET MVC Spreadsheet control | 3110 |
| See also | 3111 |
| Open and Save in Spreadsheet control | 3111 |
| Open | 3111 |
| Supported file formats | 3114 |
| Save | 3114 |
| Server Configuration | 3122 |
| Server Dependencies | 3123 |
| See Also | 3123 |
| Worksheet in Spreadsheet control | 3123 |
| Add sheet | 3123 |
| Delete sheet | 3125 |
| Rename sheet | 3125 |
| Headers | 3126 |
| Gridlines | 3126 |
| Sheet visibility | 3127 |
| See Also | 3130 |
| Cell Range in Spreadsheet control | 3130 |
| Wrap text | 3130 |
| Merge cells | 3132 |

| | |
|--|------|
| Data Validation..... | 3135 |
| Auto Fill | 3138 |
| Clear | 3141 |
| See Also | 3143 |
| Editing in Spreadsheet control..... | 3143 |
| Edit cell..... | 3143 |
| Save cell..... | 3144 |
| Cancel editing..... | 3144 |
| Limitations..... | 3145 |
| See Also | 3145 |
| Formulas in Spreadsheet control..... | 3146 |
| Usage..... | 3146 |
| Create User Defined Functions / Custom Functions..... | 3146 |
| Formula bar | 3151 |
| Named Ranges | 3151 |
| Supported Formulas..... | 3153 |
| Formula Error Dialog..... | 3156 |
| See Also | 3157 |
| Formatting in Spreadsheet Component | 3157 |
| Number Formatting | 3157 |
| Text and cell formatting..... | 3161 |
| Conditional Formatting | 3165 |
| See Also | 3168 |
| Illustrations in Spreadsheet control..... | 3168 |
| Image..... | 3168 |
| Chart..... | 3173 |
| See Also | 3178 |
| Data Binding in Spreadsheet Control..... | 3178 |
| Local data | 3178 |
| Remote data..... | 3179 |
| Cell data binding | 3182 |
| Dynamic data binding and Datasource change event | 3183 |
| See Also | 3186 |
| Filtering in Spreadsheet control..... | 3187 |
| Apply filter on UI | 3187 |

| | |
|---|------|
| Filter by criteria..... | 3187 |
| Filter by cell value | 3189 |
| Clear filter..... | 3189 |
| Clear filter on a field..... | 3189 |
| Reapply filter | 3189 |
| Known error validations..... | 3189 |
| Limitations..... | 3189 |
| See Also | 3190 |
| Sorting in Spreadsheet control | 3190 |
| Sort by cell value | 3190 |
| Data contains header..... | 3192 |
| Case sensitive sort..... | 3192 |
| Sort multiple columns | 3193 |
| Custom sort comparer | 3195 |
| Known error validations..... | 3195 |
| Limitations..... | 3195 |
| See Also | 3196 |
| Hyperlink in Spreadsheet control | 3196 |
| Insert Link..... | 3196 |
| Edit Hyperlink..... | 3196 |
| Remove Hyperlink..... | 3196 |
| How to change target attribute | 3197 |
| Limitations..... | 3199 |
| See Also | 3199 |
| Clipboard in Spreadsheet control | 3199 |
| Cut..... | 3199 |
| Copy | 3199 |
| Paste..... | 3200 |
| Prevent the paste functionality | 3202 |
| Limitations..... | 3204 |
| Selection in Spreadsheet Control..... | 3204 |
| Cell selection | 3205 |
| Row selection | 3205 |
| Column selection | 3206 |
| How to remove selection in the spreadsheet..... | 3208 |

| | |
|---|------|
| Limitations..... | 3209 |
| Scrolling in Spreadsheet control | 3209 |
| Finite Scrolling..... | 3209 |
| Virtual Scrolling..... | 3209 |
| Finite scrolling with defined rows and columns | 3210 |
| Protection in Spreadsheet Control | 3211 |
| Protect Sheet | 3212 |
| Unprotect Sheet..... | 3214 |
| Unlock the particular cells in the protected sheet..... | 3214 |
| Protect Workbook..... | 3216 |
| Unprotect Workbook..... | 3219 |
| See Also | 3219 |
| Rows and columns in Spreadsheet control..... | 3219 |
| Insert | 3219 |
| Delete..... | 3223 |
| Hide and show | 3225 |
| Size | 3227 |
| Changing text in column headers | 3230 |
| Limitations of insert and delete..... | 3230 |
| See Also | 3230 |
| Undo and Redo in Spreadsheet control..... | 3230 |
| Undo..... | 3231 |
| Redo | 3231 |
| Update custom actions in UndoRedo collection..... | 3231 |
| See Also | 3232 |
| Find and Replace in Spreadsheet control | 3232 |
| Find..... | 3232 |
| Replace..... | 3233 |
| Go to..... | 3233 |
| Limitations..... | 3235 |
| Accessibility in ASP.NET MVC Spreadsheet Control | 3235 |
| WAI-ARIA attributes..... | 3236 |
| Keyboard interaction | 3237 |
| Ensuring accessibility | 3238 |
| Keyboard Shortcuts And Navigation..... | 3238 |

| | |
|---|------|
| See Also | 3241 |
| Ribbon in Spreadsheet control | 3242 |
| Ribbon Customization | 3242 |
| See Also | 3245 |
| FreezePanels in Spreadsheet control | 3245 |
| Apply freeze panes on UI | 3245 |
| FrozenRows | 3245 |
| FrozenColumns | 3246 |
| Limitations | 3251 |
| See Also | 3251 |
| Context Menu in Spreadsheet control | 3251 |
| Context Menu Items in Row Cell | 3251 |
| Context Menu Items in Row Header / Column Header | 3251 |
| Context Menu Items in Pager | 3252 |
| Context Menu Customization | 3252 |
| See Also | 3264 |
| Cell Template in Spreadsheet Control | 3265 |
| See Also | 3267 |
| Globalization in ASP.NET MVC Spreadsheet control | 3268 |
| Localization | 3268 |
| Internationalization | 3281 |
| Right to left (RTL) | 3283 |
| See Also | 3285 |
| Use Cases | 3285 |
| Collaborative Editing | 3285 |
| How To | 3289 |
| Sort a range by custom list | 3289 |
| Print in Spreadsheet Control | 3291 |
| Mobile Responsiveness | 3303 |
| Comparison between EJ1 & EJ2 Spreadsheet features | 3304 |
| See Also | 3306 |
| Stepper | 3306 |
| Getting Started with ASP.NET MVC Stepper Control | 3306 |
| Create ASP.NET MVC application with HTML helper | 3306 |
| Install ASP.NET MVC package in the application | 3306 |

| | |
|---|------|
| Add namespace..... | 3307 |
| Add stylesheet and script resources | 3307 |
| Register Syncfusion script manager | 3307 |
| Add ASP.NET MVC Stepper control..... | 3308 |
| Adding steps..... | 3308 |
| Configure icon and label | 3308 |
| Steps in ASP.NET MVC Stepper control | 3311 |
| Adding steps..... | 3311 |
| Optional steps | 3313 |
| Disabling steps | 3316 |
| Setting active step..... | 3318 |
| Step status..... | 3320 |
| Step styling..... | 3323 |
| Step validation | 3325 |
| Step types in ASP.NET MVC Stepper control | 3328 |
| Default type | 3328 |
| Label type..... | 3330 |
| Indicator type..... | 3333 |
| Orientation in ASP.NET MVC Stepper control | 3334 |
| Horizontal..... | 3334 |
| Vertical | 3336 |
| Linear flow in ASP.NET MVC Stepper control | 3339 |
| Steps validation in ASP.NET MVC Stepper control | 3341 |
| Template in ASP.NET MVC Stepper control..... | 3344 |
| Tooltip in ASP.NET MVC Stepper control..... | 3346 |
| Tooltip template | 3349 |
| Animation in ASP.NET MVC Stepper control | 3351 |
| Globalization in ASP.NET MVC Stepper control..... | 3352 |
| Localization | 3352 |
| RTL..... | 3355 |
| Accessibility in ASP.NET MVC Stepper control | 3357 |
| Keyboard interaction | 3357 |
| ARIA attribute | 3358 |
| Events..... | 3358 |
| Created..... | 3358 |

| | |
|---|------|
| StepChanged | 3358 |
| StepChanging | 3359 |
| StepClick..... | 3359 |
| BeforeStepRender..... | 3360 |
| Stock Chart..... | 3360 |
| Getting Started with ASP.NET MVC Stock Chart Control..... | 3360 |
| Prerequisites | 3361 |
| Create ASP.NET MVC application with HTML helper..... | 3361 |
| Install ASP.NET MVC package in the application | 3361 |
| Add namespace..... | 3361 |
| Add stylesheet and script resources | 3361 |
| Register Syncfusion script manager..... | 3362 |
| Add ASP.NET MVC Stock Chart control..... | 3362 |
| Populate Stock Chart With Data | 3362 |
| Working with Data | 3363 |
| Local Data..... | 3363 |
| See Also | 3364 |
| Stock Chart Dimensions | 3364 |
| Size for Container..... | 3364 |
| Size for Stock Chart | 3365 |
| Axis types | 3367 |
| DateTime axis..... | 3367 |
| DateTimeCategory axis | 3368 |
| Logarithmic axis | 3369 |
| See also | 3370 |
| Axis Customization..... | 3370 |
| Axis Crossing | 3370 |
| Title | 3371 |
| Tick Lines Customization..... | 3372 |
| Grid Lines Customization | 3372 |
| Multiple Axis | 3373 |
| Inversed Axis | 3374 |
| Opposed Position | 3375 |
| Stock Chart Series Types | 3376 |
| Trendlines | 3377 |

| | |
|--|------|
| Linear..... | 3377 |
| Exponential | 3377 |
| Logarithmic | 3377 |
| Polynomial | 3377 |
| Power | 3378 |
| Moving Average | 3378 |
| Customization of Trendline..... | 3379 |
| Technical Indicators | 3379 |
| Accumulation Distribution | 3380 |
| Average True Range (ATR) | 3380 |
| Exponential Moving Average (EMA) | 3380 |
| Momentum | 3380 |
| Moving Average Convergence Divergence (MACD) | 3380 |
| Relative Strength Index (RSI)..... | 3380 |
| Simple Moving Average (SMA) | 3380 |
| Stochastic | 3380 |
| Triangular Moving Average (TMA)..... | 3380 |
| Bollinger Band | 3380 |
| Tooltip | 3382 |
| Default tooltip..... | 3382 |
| Format the tooltip..... | 3382 |
| Position the tooltip | 3383 |
| Customize the appearance of the tooltip | 3384 |
| Add Crosshair | 3385 |
| Tooltip for axis | 3386 |
| Customization | 3387 |
| Add Trackball..... | 3388 |
| Legend..... | 3389 |
| Position and Alignment..... | 3389 |
| Legend Alignment | 3391 |
| Customization | 3392 |
| Legend Size..... | 3393 |
| Legend Item Size | 3393 |
| Collapsing Legend Item | 3394 |
| Legend Title | 3395 |

| | |
|--|------|
| Period selector | 3396 |
| Periods | 3396 |
| Visibility of period selector | 3398 |
| Range selector..... | 3399 |
| Selecting Range..... | 3399 |
| Print and Export | 3400 |
| Disable Export and print | 3400 |
| Appearance | 3401 |
| Stock Chart Title | 3401 |
| Title Customizations..... | 3402 |
| Stock Chart Theme | 3403 |
| See Also | 3404 |
| Stock Events | 3404 |
| Accessibility in ASP.NET MVC Stock chart component | 3409 |
| WAI-ARIA attributes..... | 3410 |
| Keyboard interaction | 3411 |
| Ensuring accessibility | 3411 |
| See also | 3411 |
| Switch..... | 3411 |
| Getting Started with ASP.NET MVC Switch Control..... | 3411 |
| Prerequisites | 3411 |
| Create ASP.NET MVC application with HTML helper..... | 3411 |
| Install ASP.NET MVC package in the application | 3412 |
| Add namespace..... | 3412 |
| Add stylesheet and script resources | 3412 |
| Register Syncfusion script manager | 3412 |
| Add ASP.NET MVC Switch control..... | 3413 |
| Set text on Switch | 3413 |
| See also | 3413 |
| Accessibility in Switch Button Controls..... | 3413 |
| WAI-ARIA attributes..... | 3414 |
| Keyboard interaction | 3414 |
| Ensuring accessibility | 3415 |
| See also | 3415 |
| Styles and Appearances | 3415 |

| | |
|--|------|
| How To | 3415 |
| Change Size | 3415 |
| Customize the appearance of a Switch..... | 3416 |
| Enable ripple for Switch label | 3419 |
| Enable RTL..... | 3420 |
| Set disabled state..... | 3420 |
| Submit name and value in form..... | 3421 |
| Tabs..... | 3422 |
| Getting Started with ASP.NET MVC Tab Control | 3422 |
| Prerequisites | 3422 |
| Create ASP.NET MVC application with HTML helper..... | 3422 |
| Install ASP.NET MVC package in the application | 3422 |
| Add namespace..... | 3422 |
| Add stylesheet and script resources | 3423 |
| Register Syncfusion script manager | 3423 |
| Add ASP.NET MVC Tab control | 3423 |
| Initialize the Tab using JSON items collection..... | 3425 |
| Initialize the Tab using HTML elements | 3426 |
| See also | 3427 |
| Responsive Modes | 3428 |
| Scrollable..... | 3428 |
| Popup..... | 3430 |
| See Also | 3433 |
| Header..... | 3433 |
| Styles | 3433 |
| Icon positions | 3435 |
| See Also | 3440 |
| Orientation..... | 3440 |
| Localization | 3444 |
| Loading translations..... | 3444 |
| Drag and drop items | 3445 |
| Drag and drop item between tabs..... | 3447 |
| Drag and drop items to external source | 3451 |
| Drag and drop items from external source..... | 3455 |
| Accessibility ASP.NET MVC Tab control | 3459 |

| | |
|--|------|
| ARIA attributes | 3460 |
| Keyboard interaction | 3461 |
| Ensuring accessibility | 3461 |
| See also | 3461 |
| CSS Structure..... | 3462 |
| Customizing Tab | 3462 |
| Customizing the Tab items..... | 3462 |
| Customizing Tab's header | 3462 |
| Customizing Tab's header icon | 3462 |
| Customizing Tab's content..... | 3463 |
| Customizing the hover state of Tab control..... | 3463 |
| Customizing selected item of Tab control | 3463 |
| How To | 3464 |
| Load content through Ajax | 3464 |
| Prevent content swipe selection..... | 3465 |
| Customize selected tab styles..... | 3466 |
| Create wizard using Tab..... | 3469 |
| Load tab with DataSource..... | 3480 |
| Add nested Tabs..... | 3481 |
| Set state persistence of the Tab Control | 3484 |
| Set custom animation | 3486 |
| Load Tab items dynamically..... | 3488 |
| Create collapsible Tabs | 3490 |
| Customize Tab content height | 3493 |
| How to customize Tab scrollStep..... | 3495 |
| Populate Tab items and their content through ViewBag | 3496 |
| Render the Tab items using content template | 3498 |
| Load the content as partial view to Tab | 3500 |
| How to prevent reorder active tab while selecting inside popup | 3503 |
| How to find whether the tab is selected programmatically or user interaction | 3506 |
| Enabling tab key navigation in Tabs..... | 3510 |
| Migration from Essential JS 1..... | 3512 |
| Accessibility..... | 3512 |
| AjaxSettings..... | 3512 |
| Animation..... | 3513 |

| | |
|--|------|
| Header..... | 3514 |
| Items | 3514 |
| Common..... | 3516 |
| TextBox | 3517 |
| Getting Started with ASP.NET MVC TextBox Control | 3517 |
| Prerequisites | 3517 |
| Create ASP.NET MVC application with HTML helper | 3517 |
| Install ASP.NET MVC package in the application | 3518 |
| Add namespace..... | 3518 |
| Add stylesheet and script resources | 3518 |
| Register Syncfusion script manager | 3518 |
| Add ASP.NET MVC TextBox control | 3519 |
| Adding icons to the TextBox | 3519 |
| Floating label..... | 3520 |
| See also | 3521 |
| Groups..... | 3521 |
| With icon and floating label | 3522 |
| With clear button and floating label | 3524 |
| Floating label without required attribute | 3525 |
| Multi-line input with floating label | 3527 |
| See Also | 3528 |
| Sizing | 3528 |
| Validation..... | 3530 |
| Multiline TextBox..... | 3531 |
| Create multiline textbox | 3531 |
| Implementing floating label | 3532 |
| Auto resizing | 3534 |
| Disable resizing | 3535 |
| Limit the text length..... | 3536 |
| Count characters..... | 3537 |
| Accessibility in ASP.NET MVC Textbox component | 3538 |
| WAI-ARIA attributes..... | 3539 |
| Ensuring accessibility | 3539 |
| See also | 3539 |
| Style and appearance in TextBox Component..... | 3539 |

| | |
|--|------|
| Customizing the appearance of TextBox wrapper element | 3539 |
| Customizing the TextBox placeholder | 3540 |
| How To | 3540 |
| Set the rounded corner..... | 3540 |
| Set the disabled state..... | 3541 |
| Set the read-only TextBox..... | 3542 |
| Add TextBox programmatically..... | 3543 |
| Add floating label to TextBox programmatically | 3545 |
| Change the floating label color of the TextBox..... | 3547 |
| Change the color of the TextBox based on its value..... | 3549 |
| Add floating label to read-only textbox | 3550 |
| ASP.NET MVC textbox control | 3553 |
| Normal textbox | 3553 |
| Floating label textbox..... | 3553 |
| TimePicker..... | 3554 |
| Getting Started with ASP.NET MVC TimePicker Control..... | 3554 |
| Prerequisites | 3554 |
| Create ASP.NET MVC application with HTML helper..... | 3554 |
| Install ASP.NET MVC package in the application | 3554 |
| Add namespace..... | 3554 |
| Add stylesheet and script resources | 3555 |
| Register Syncfusion script manager | 3555 |
| Add ASP.NET MVC TimePicker control | 3555 |
| Setting the value within min and max time | 3556 |
| Setting the time format | 3557 |
| See also | 3558 |
| Time Range | 3558 |
| Time Range customization using two TimePicker components | 3558 |
| Enable the Masked Input..... | 3560 |
| Configure Mask Placeholder | 3561 |
| Globalization | 3563 |
| Right-To-Left | 3565 |
| Strict mode in TimePicker Control | 3566 |
| Accessibility..... | 3567 |
| WAI-ARIA attributes..... | 3569 |

| | |
|--|------|
| Keyboard Interaction | 3569 |
| Ensuring accessibility | 3570 |
| See also | 3570 |
| Style and appearance in TimePicker Component..... | 3570 |
| Customizing the appearance of TimePicker wrapper element | 3570 |
| Customizing the TimePicker icon element..... | 3570 |
| Customizing the TimePicker popup | 3571 |
| Customizing the TimePicker popup content..... | 3571 |
| Full screen mode support in mobiles and tablets..... | 3571 |
| How To | 3573 |
| CSS customization | 3573 |
| Client side validation using FormValidator | 3574 |
| Render TimePickerFor | 3575 |
| Toast..... | 3577 |
| Getting Started with ASP.NET MVC Toast Control..... | 3577 |
| Prerequisites | 3577 |
| Create ASP.NET MVC application with HTML helper..... | 3577 |
| Install ASP.NET MVC package in the application | 3577 |
| Add namespace..... | 3577 |
| Add stylesheet and script resources | 3577 |
| Register Syncfusion script manager | 3578 |
| Add ASP.NET MVC Toast control | 3578 |
| See also | 3579 |
| Configuring Options | 3579 |
| Title and content template | 3579 |
| Specifying custom target | 3579 |
| Close button | 3580 |
| Progress bar | 3580 |
| Newest on top..... | 3580 |
| Width and height | 3582 |
| See Also | 3584 |
| Positions..... | 3584 |
| Predefined..... | 3584 |
| Custom | 3584 |
| See Also | 3590 |

| | |
|--|------|
| Action Buttons | 3590 |
| See Also | 3591 |
| Time out | 3591 |
| Static toast | 3593 |
| See Also | 3594 |
| Template | 3594 |
| See Also | 3597 |
| Animations | 3597 |
| Toast Utility Services | 3599 |
| Show Toast with predefined types | 3599 |
| Show Toast with ToastModel | 3601 |
| CSS structures | 3601 |
| Customizing the toast title | 3602 |
| Customizing the toast content | 3602 |
| Customizing the toast icon | 3602 |
| Customizing the toast background | 3602 |
| Accessibility | 3603 |
| WAI-ARIA attributes | 3604 |
| Ensuring accessibility | 3604 |
| See also | 3605 |
| How To | 3605 |
| Prevent duplicate toast display | 3605 |
| Restrict the maximum toast to show | 3606 |
| Customize progress bar theme and sizing | 3607 |
| Play an audio before open the toast | 3609 |
| Show different types of toast | 3610 |
| Show multiple toasts in various positions | 3611 |
| Close the toast with click/tap | 3612 |
| Add dynamic template | 3613 |
| Prevent toast close with mobile swipe | 3614 |
| ToolBar | 3615 |
| Getting Started with ASP.NET MVC Toolbar Control | 3615 |
| Prerequisites | 3615 |
| Create ASP.NET MVC application with HTML helper | 3615 |
| Install ASP.NET MVC package in the application | 3615 |

| | |
|--|------|
| Add namespace..... | 3616 |
| Add stylesheet and script resources | 3616 |
| Register Syncfusion script manager | 3616 |
| Add ASP.NET MVC Toolbar control..... | 3616 |
| Render the Toolbar items using content template..... | 3617 |
| See also | 3618 |
| Item Configuration in ASP.NET MVC Toolbar control..... | 3618 |
| Button | 3618 |
| Separator..... | 3619 |
| Input..... | 3619 |
| See Also | 3622 |
| Responsive Mode..... | 3622 |
| Scrollable..... | 3622 |
| Popup..... | 3625 |
| Template Configuration | 3628 |
| Popup customization | 3629 |
| Integrate menu component..... | 3631 |
| Accessibility in ASP.NET MVC Toolbar control..... | 3634 |
| ARIA attributes..... | 3635 |
| Keyboard interaction | 3635 |
| Ensuring accessibility | 3636 |
| See also | 3636 |
| CSS Structure in ASP.NET MVC Toolbar Component | 3636 |
| Customizing Toolbar | 3636 |
| Customizing the Toolbar items | 3636 |
| Customizing Toolbar's item icon..... | 3637 |
| Customizing the hover state of Toolbar control | 3637 |
| Customizing selected item of Toolbar control..... | 3637 |
| Set command customization | 3637 |
| Set Essential JS 2 Tooltip to the commands..... | 3638 |
| Set item-wise custom template | 3639 |
| Add Toggle Button in ToolBar Control | 3640 |
| How to customize toolbar scrollStep..... | 3642 |
| ToolTip | 3649 |
| Getting Started with ASP.NET MVC Tooltip Control | 3649 |

| | |
|---|------|
| Prerequisites | 3649 |
| Create ASP.NET MVC application with HTML helper | 3649 |
| Install ASP.NET MVC package in the application | 3649 |
| Add namespace..... | 3649 |
| Add stylesheet and script resources | 3650 |
| Register Syncfusion script manager | 3650 |
| Add ASP.NET MVC Tooltip control | 3650 |
| Initialize Tooltip within a container | 3651 |
| See also | 3652 |
| Setting Dimension in Tooltip Control..... | 3652 |
| Height and width..... | 3652 |
| Content in Tooltip Control | 3654 |
| Template content..... | 3654 |
| Dynamic content via Fetch..... | 3655 |
| Tooltip Positioning in Tooltip Control | 3657 |
| Tip pointer positioning..... | 3660 |
| Dynamic positioning..... | 3661 |
| Mouse Trailing | 3663 |
| Setting offset values..... | 3664 |
| Open Mode in Tooltip Control | 3664 |
| Custom open mode..... | 3667 |
| Sticky mode..... | 3668 |
| Open/Close Tooltip with delay | 3669 |
| Animation in Tooltip Control | 3670 |
| Animation effects..... | 3671 |
| Animating via open/close methods | 3671 |
| Apply transition..... | 3672 |
| Customization in Tooltip Control | 3674 |
| Tip pointer customization | 3674 |
| Tooltip customization | 3675 |
| Styles and Appearance..... | 3677 |
| Customizing the tooltip..... | 3677 |
| Customizing the tooltip popup | 3678 |
| Customizing the tooltip content | 3678 |
| Customizing the tooltip arrow tip..... | 3678 |

| | |
|--|------|
| Customizing the tooltip inner tip | 3679 |
| Customizing the tooltip outer tip..... | 3679 |
| Accessibility in ASP.NET MVC Tooltip component..... | 3680 |
| WAI-ARIA attributes..... | 3681 |
| Keyboard interaction | 3682 |
| Ensuring accessibility | 3682 |
| See also | 3682 |
| Migration from Essential JS 1..... | 3682 |
| How To | 3684 |
| Show Tooltip on disabled elements and disable tooltip..... | 3684 |
| Dynamic Tooltip content with HTML elements | 3684 |
| Custom Tooltip with dynamic HTML..... | 3686 |
| Tooltip open or display modes | 3687 |
| Fancy Tooltip Customization..... | 3689 |
| Display Tooltip on SVG and canvas elements | 3695 |
| Dynamic Tooltip content..... | 3698 |
| Create and show Tooltip on multiple targets | 3701 |
| Tree Grid | 3704 |
| Getting Started with ASP.NET MVC Tree Grid Control | 3704 |
| Prerequisites | 3704 |
| Create ASP.NET MVC application with HTML helper | 3704 |
| Install ASP.NET MVC package in the application | 3704 |
| Add namespace..... | 3704 |
| Add stylesheet and script resources | 3705 |
| Register Syncfusion script manager | 3705 |
| Add ASP.NET MVC Tree Grid control | 3705 |
| Defining Row Data | 3705 |
| Defining Columns..... | 3708 |
| Enable Paging..... | 3711 |
| Enable Sorting..... | 3713 |
| Enable Filtering | 3716 |
| Data binding in ASP.NET MVC Tree Grid Component..... | 3719 |
| Binding with ajax..... | 3719 |
| Handling expandStateMapping | 3720 |
| Immutable Mode | 3721 |

| | |
|--|------|
| Limitations..... | 3722 |
| Columns in ASP.NET MVC Tree Grid Component | 3722 |
| Format..... | 3723 |
| Lock columns..... | 3724 |
| Column type..... | 3725 |
| Checkbox column..... | 3725 |
| Controlling Tree Grid actions | 3726 |
| Show/hide columns by external button | 3726 |
| ValueAccessor | 3727 |
| How to render boolean values as checkbox | 3729 |
| Rows in ASP.NET MVC Tree Grid Component..... | 3730 |
| Customize rows..... | 3730 |
| Styling alternate rows | 3731 |
| Cell in ASP.NET MVC Tree Grid Component | 3731 |
| Customize cell styles | 3731 |
| Editing in ASP.NET MVC Tree Grid Component | 3736 |
| Toolbar with edit option | 3737 |
| Adding row position..... | 3737 |
| Confirmation messages..... | 3738 |
| Default column values on add new..... | 3739 |
| Disable editing for particular column | 3740 |
| Troubleshoot: Editing works only for first row | 3741 |
| Sorting..... | 3741 |
| Initial sort | 3742 |
| Sorting events | 3742 |
| Touch interaction | 3744 |
| Filtering in ASP.NET MVC Tree Grid Component | 3746 |
| Filter hierarchy modes | 3746 |
| Initial filter..... | 3747 |
| Filter operators | 3748 |
| Search..... | 3749 |
| Initial search..... | 3749 |
| Search operators..... | 3750 |
| Search by external button..... | 3750 |
| Search specific columns | 3751 |

| | |
|--|------|
| Paging..... | 3752 |
| Page Size Mode | 3752 |
| Template | 3753 |
| Pager with Page Size Dropdown | 3754 |
| How to render Pager at the Top of the TreeGrid..... | 3756 |
| Selection in ASP.NET MVC Tree Grid Component | 3757 |
| Selection mode | 3758 |
| Aggregates in ASP.NET MVC Tree Grid Component | 3758 |
| Built-in aggregate types | 3758 |
| Child aggregate | 3759 |
| Context menu in ASP.NET MVC Tree Grid Component | 3760 |
| Custom context menu items..... | 3761 |
| Enable and disable context menu items dynamically..... | 3763 |
| ToolBar in ASP.NET MVC Tree Grid Component..... | 3765 |
| Enable/disable toolbar items..... | 3765 |
| Globalization in ASP.NET MVC Tree Grid control | 3767 |
| Localization | 3767 |
| Internationalization..... | 3771 |
| Right to left (RTL) | 3773 |
| See Also | 3774 |
| Scrolling..... | 3774 |
| Set width and height..... | 3775 |
| Responsive with parent container | 3775 |
| Scroll to selected row..... | 3776 |
| Frozen rows and columns | 3777 |
| Frozen rows and columns | 3777 |
| Tree Grid Virtualization..... | 3779 |
| Row Virtualization..... | 3779 |
| Column Virtualization | 3780 |
| Limitations for Virtualization | 3782 |
| Tree Grid Infinite scrolling..... | 3783 |
| InitialBlocks | 3783 |
| Cache Mode | 3784 |
| Limitations for Infinite Scrolling..... | 3785 |
| State Persistence..... | 3785 |

| | |
|---|------|
| Get or set localStorage value | 3785 |
| Print..... | 3786 |
| Page setup..... | 3786 |
| Print using an external button | 3787 |
| Print the visible page..... | 3787 |
| Print large number of columns | 3788 |
| Show or Hide columns while Printing | 3788 |
| Limitations of Printing Large Data..... | 3790 |
| Adaptive View in ASP.NET MVC Tree Grid Component..... | 3790 |
| Render adaptive dialogs..... | 3790 |
| Loading Animation in ASP.NET MVC Tree Grid Component..... | 3791 |
| PDF Export in ASP.NET MVC Tree Grid Component | 3791 |
| Excel Export in ASP.NET MVC Tree Grid Component | 3792 |
| Persist collapsed state | 3793 |
| Clipboard..... | 3794 |
| Copy to clipboard by external buttons | 3795 |
| Copy Hierarchy Modes..... | 3796 |
| AutoFill | 3798 |
| Paste..... | 3799 |
| Accessibility in Tree Grid control | 3800 |
| WAI-ARIA attributes..... | 3801 |
| Keyboard interaction | 3801 |
| Ensuring accessibility | 3803 |
| See also | 3803 |
| Styling..... | 3803 |
| TreeMap..... | 3804 |
| Getting Started with ASP.NET MVC TreeMap Control..... | 3804 |
| Prerequisites | 3804 |
| Create ASP.NET MVC application with HTML helper..... | 3804 |
| Install ASP.NET MVC package in the application | 3804 |
| Add namespace..... | 3804 |
| Add script resources | 3805 |
| Register Syncfusion script manager | 3805 |
| Add ASP.NET MVC TreeMap control..... | 3805 |
| Data Binding..... | 3806 |

| | |
|---|------|
| Populate data | 3807 |
| Layout..... | 3811 |
| Types of layout..... | 3811 |
| Right-to-left rendering | 3817 |
| Legend with Rtl support..... | 3817 |
| Tooltip with Rtl support | 3818 |
| Treemap Item Rendering Direction | 3820 |
| Leaf Item | 3825 |
| Leaf label | 3825 |
| Item gap | 3830 |
| Group path | 3831 |
| Group gap..... | 3833 |
| Header format and Alignment | 3835 |
| Header height and style | 3836 |
| Header template and position | 3838 |
| Color Mapping..... | 3840 |
| Range color mapping | 3840 |
| Equal color mapping | 3842 |
| Desaturation color mapping | 3843 |
| Palette color mapping..... | 3845 |
| Desaturation with multiple colors | 3846 |
| Color for items excluded from color mapping | 3847 |
| Bind the colors to the items from data source | 3849 |
| Data Label | 3850 |
| Format..... | 3850 |
| Template | 3852 |
| InterSectAction | 3853 |
| Legend..... | 3854 |
| Position and alignment | 3854 |
| Legend mode..... | 3858 |
| Legend size | 3861 |
| Legend for items excluded from color mapping | 3864 |
| Hide desired legend items | 3866 |
| Hide legend items based data source value | 3867 |
| Bind legend item text from data source | 3869 |

| | |
|--|------|
| Hide duplicate legend items | 3870 |
| Legend Responsiveness | 3871 |
| Drill-down | 3873 |
| Perform drill-down action..... | 3873 |
| On-demand data loading | 3875 |
| Breadcrumb support..... | 3877 |
| Tooltip | 3879 |
| Default tooltip..... | 3879 |
| Format tooltip | 3880 |
| Tooltip template | 3881 |
| Selection and Highlight | 3882 |
| Selection..... | 3882 |
| Highlight | 3884 |
| Print and Export | 3886 |
| Print..... | 3886 |
| Export..... | 3888 |
| Accessibility in ASP.NET MVC TreeMap component | 3891 |
| WAI-ARIA attributes..... | 3891 |
| Screen reading in TreeMap..... | 3892 |
| Ensuring accessibility | 3892 |
| See also | 3892 |
| Internationalization..... | 3892 |
| Globalization | 3892 |
| Migration from Essential JS 1..... | 3893 |
| Data Binding..... | 3893 |
| Appearance | 3894 |
| Leaf Items..... | 3895 |
| Legend..... | 3896 |
| Levels..... | 3898 |
| Selection..... | 3900 |
| Hightlight..... | 3900 |
| Range ColorMapping..... | 3901 |
| Desaturation ColorMapping | 3902 |
| Tooltip | 3903 |
| Drilldown..... | 3904 |

| | |
|---|------|
| Methods..... | 3904 |
| Events..... | 3905 |
| How To | 3906 |
| How To | 3906 |
| TreeView | 3917 |
| Getting Started with ASP.NET MVC TreeView Control | 3917 |
| Prerequisites | 3917 |
| Create ASP.NET MVC application with HTML helper | 3917 |
| Install ASP.NET MVC package in the application | 3917 |
| Add namespace..... | 3917 |
| Add stylesheet and script resources | 3917 |
| Register Syncfusion script manager | 3918 |
| Add ASP.NET MVC TreeView control..... | 3918 |
| Binding data source | 3918 |
| Data Binding in Treeview Control | 3922 |
| Local data | 3922 |
| Remote data..... | 3927 |
| CheckBox..... | 3929 |
| Checked nodes..... | 3932 |
| See Also | 3936 |
| Node Editing in Treeview Control | 3936 |
| See Also | 3939 |
| Multi Selection in TreeView Control | 3939 |
| Selected nodes | 3943 |
| Drag and Drop | 3946 |
| Multiple-node drag and drop..... | 3950 |
| See Also | 3953 |
| Template | 3953 |
| See Also | 3955 |
| Accessibility in ASP.NET MVC TreeView component..... | 3955 |
| WAI-ARIA attributes..... | 3956 |
| Keyboard interaction | 3957 |
| Ensuring accessibility | 3957 |
| See also | 3957 |
| How To | 3958 |

| | |
|---|------|
| Customize the expand and collapse icons | 3958 |
| Process the tree node operations using context menu | 3962 |
| Check/uncheck the checkbox on clicking the tree node text | 3967 |
| Validate the text when renaming the tree node | 3971 |
| Customize the tree nodes based on levels | 3975 |
| Restrict the drag-and-drop for particular tree nodes | 3979 |
| Migration from Essential JS 1..... | 3983 |
| Add nodes | 3983 |
| Common..... | 3984 |
| CheckBox..... | 3994 |
| Drag and Drop..... | 3997 |
| Expand/Collapse nodes..... | 3999 |
| Node Editing..... | 4002 |
| Node Selection | 4002 |
| Template | 4006 |
| Uploader | 4006 |
| Getting Started with ASP.NET MVC Uploader Control | 4006 |
| Prerequisites | 4006 |
| Create ASP.NET MVC application with HTML helper..... | 4006 |
| Install ASP.NET MVC package in the application | 4006 |
| Add namespace..... | 4007 |
| Add stylesheet and script resources | 4007 |
| Register Syncfusion script manager | 4007 |
| Adding drop area | 4008 |
| Configure asynchronous settings..... | 4008 |
| Handle success and failed upload | 4009 |
| See also | 4009 |
| Asynchronous upload | 4010 |
| Multiple file upload..... | 4010 |
| Single file upload..... | 4010 |
| Save action | 4011 |
| Remove action | 4013 |
| Auto upload..... | 4015 |
| Sequential upload | 4016 |
| Preload files..... | 4016 |

| | |
|---|------|
| Adding additional HTTP headers with upload action..... | 4018 |
| See Also | 4019 |
| Chunk Upload..... | 4019 |
| Additional configurations..... | 4020 |
| Resumable upload | 4021 |
| Cancel upload..... | 4022 |
| Server-Side configurations..... | 4023 |
| File source | 4027 |
| Paste to upload | 4027 |
| Directory upload | 4028 |
| Drag and drop | 4030 |
| See Also | 4032 |
| Drag and drop in Uploader Control | 4032 |
| Custom drop area | 4033 |
| Customize drop area | 4033 |
| Validation in Uploader Control | 4034 |
| File type..... | 4034 |
| File size..... | 4035 |
| Maximum files count | 4036 |
| Duplicate files..... | 4037 |
| See Also | 4038 |
| Forms Support (Synchronous Upload)..... | 4038 |
| Template | 4043 |
| File list template..... | 4043 |
| Custom template | 4050 |
| See Also | 4056 |
| Localization in Uploader Control | 4056 |
| Accessibility..... | 4058 |
| Keyboard interaction | 4059 |
| Ensuring accessibility | 4060 |
| See also | 4060 |
| Style and appearance in Uploader Component..... | 4060 |
| Customizing the appearance of File Upload wrapper element | 4060 |
| Customizing the File Upload browse button | 4061 |
| Customizing the File Upload content..... | 4061 |

| | |
|---|------|
| Customizing the uploaded file container in File Upload | 4061 |
| See Also | 4062 |
| How To | 4062 |
| Hide default drop area | 4062 |
| Preview images before uploading | 4063 |
| Achieve invisible upload..... | 4063 |
| Customize progressbar | 4064 |
| Sort the selected files..... | 4065 |
| Get the total size of selected files..... | 4066 |
| Customize button with HTML element..... | 4067 |
| Add confirm dialog to remove the files | 4068 |
| Add additional data on upload | 4069 |
| Validate image/* on drop | 4069 |
| Determine whether uploader has file input (required validation) | 4070 |
| Achieve file upload programmatically | 4074 |
| Check file size before uploading it | 4075 |
| Check the MIME type of file before upload it..... | 4076 |
| Trigger click event of input file from external button..... | 4077 |
| Open and edit the uploaded files | 4078 |
| Resize images before uploading it to the server..... | 4080 |
| Convert image into binary format after uploading..... | 4086 |
| Migration from Essential JS 1..... | 4087 |
| Accessibility | 4087 |
| File list | 4087 |
| File selection | 4088 |
| Upload action | 4089 |
| Chunk upload | 4091 |
| Remove action | 4092 |
| Buttons..... | 4092 |
| Drag and drop | 4093 |
| Common..... | 4093 |

Kanban

Getting Started with ASP.NET MVC Kanban Control

This section briefly explains about how to include [ASP.NET MVC Kanban](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

<add namespace="Syncfusion.EJ2"/>

</namespaces>

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{ site.ej2version
  }/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

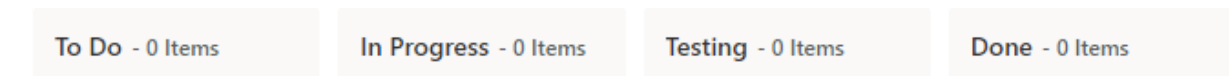
Add ASP.NET MVC Kanban control

Now, add the Syncfusion ASP.NET MVC Kanban control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().Kanban("kanban").Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Kanban control will be rendered in the default web browser.



Populating cards

To populate the empty Kanban with cards, define the list or remote data using the [DataSource](#) property. To define [DataSource](#), the mandatory fields in the list should be relevant to [KeyField](#). In the following example, you can see the cards defined with default fields such as ID, Summary, and Status.

CSHTML

```
@model List<KanbanSample.Controllers.KanbanDataModels>
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)Model).Columns(col =>
```

```

        {
            col.HeaderText("To Do").KeyField("Open").Add();
            col.HeaderText("In Progress").KeyField("InProgress").Add();
            col.HeaderText("Testing").KeyField("Testing").Add();
            col.HeaderText("Done").KeyField("Close").Add();
        }).CardSettings(card => {
            card.ContentField("Summary").HeaderField("Id");
        }).Render()
    }
}

```

HOMECONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View(KanbanDataModels.KanbanTasks());
    }
}

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public static List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title = "Task - 29001", Status = "Open", Summary = "Analyze the new requirements gathered from the customer.", Type = "Story", Priority = "Low", Tags = "Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title = "Task - 29002", Status = "InProgress", Summary = "Improve application performance", Type = "Improvement", Priority = "Normal", Tags = "Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title = "Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the customer to get new requirements.", Type = "Others", Priority = "Critical", Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2, Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title = "Task - 29004", Status = "InProgress", Summary = "Fix the issues reported in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title = "Task - 29005", Status = "Review", Summary = "Fix the issues reported by the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate = 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title = "Task -
29007", Status = "Validate", Summary = "Validate new requirements", Type =
"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title = "Task -
29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title = "Task -
29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title = "Task -
29011", Status = "Validate", Summary = "Validate the issues reported by the
customer.", Type = "Story", Priority = "High", Tags = "Validation,Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title = "Task -
29015", Status = "Open", Summary = "Show the retrieved data from the server
in grid control.", Type = "Story", Priority = "High", Tags = "Database,SQL",
Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title = "Task -
29016", Status = "InProgress", Summary = "Fix cannot open user's default
database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title = "Task -
29017", Status = "Review", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title = "Task -
29018", Status = "Close", Summary = "Analyze SQL server 2008 connection.",
Type = "Story", Priority = "Release Breaker", Tags = "Grid,Sql", Estimate =
2, Assignee = "Andrew Fuller", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title = "Task -
29019", Status = "Validate", Summary = "Validate databinding issues.", Type
= "Story", Priority = "Low", Tags = "Validation", Estimate = 1.5, Assignee =
"Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title = "Task -
29020", Status = "Close", Summary = "Analyze grid control.", Type = "Story",
Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee = "Margaret
hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title = "Task -
29021", Status = "Close", Summary = "Stored procedure for initial data
binding of the grid.", Type = "Others", Priority = "Release Breaker", Tags =
"Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId = 6, Color
= "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title = "Task -
29022", Status = "Close", Summary = "Analyze stored procedures.", Type =
"Story", Priority = "Release Breaker", Tags = "Procedures", Estimate = 5.5,
Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title = "Task -
29023", Status = "Validate", Summary = "Validate editing issues.", Type =
"Story", Priority = "Critical", Tags = "Editing", Estimate = 1, Assignee =
"Nancy Davloio", RankId = 1, Color = "#8b447a" });

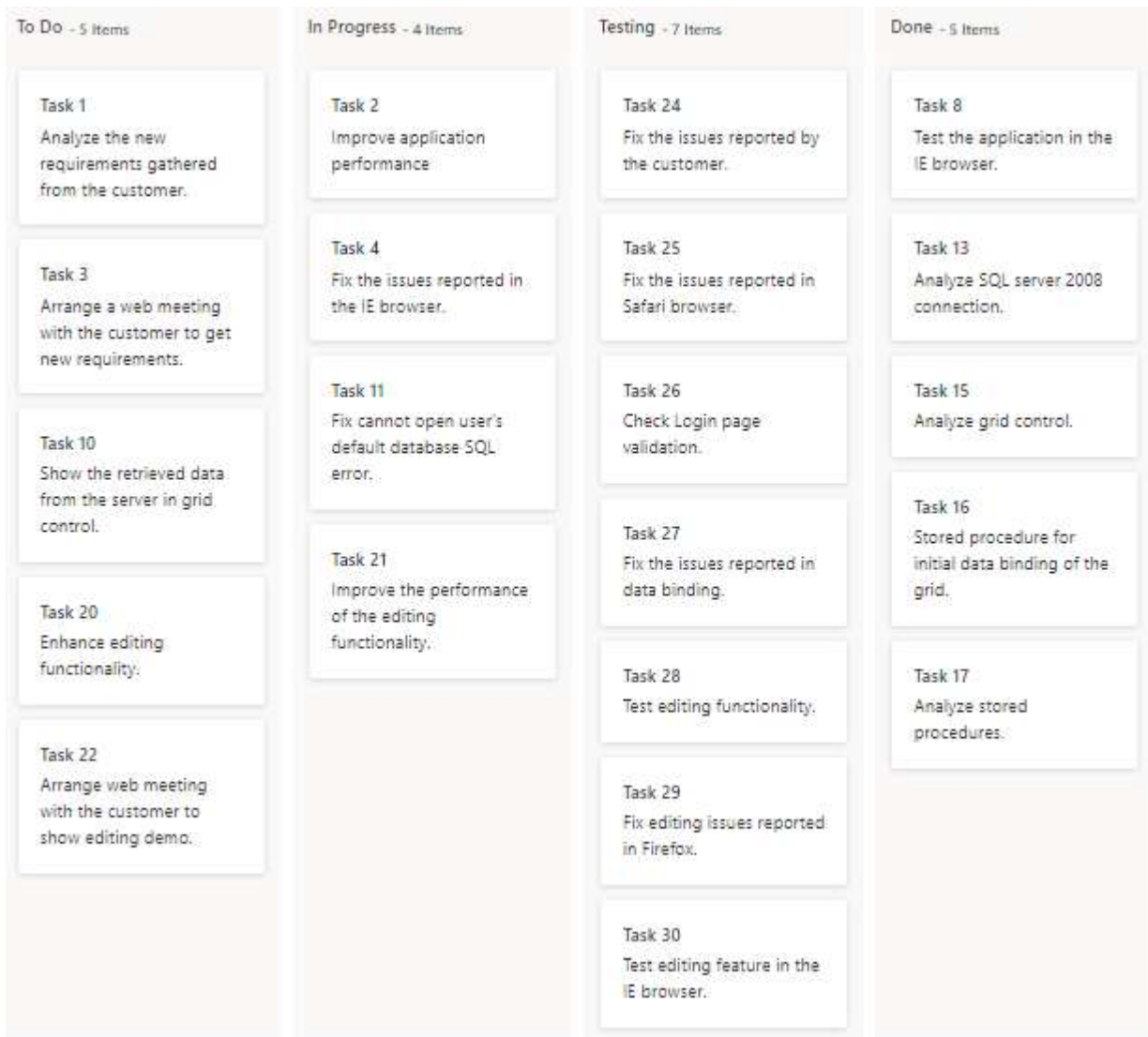
```



```

TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title = "Task -
29024", Status = "Review", Summary = "Test editing functionality.", Type =
"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title = "Task -
29025", Status = "Open", Summary = "Enhance editing functionality.", Type =
"Improvement", Priority = "Low", Tags = "Editing", Estimate = 3.5, Assignee
= "Andrew Fuller", RankId = 5, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title = "Task -
29026", Status = "InProgress", Summary = "Improve the performance of the
editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title = "Task -
29027", Status = "Open", Summary = "Arrange web meeting with the customer to
show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title = "Task -
29029", Status = "Review", Summary = "Fix the editing issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Editing,Fix", Estimate =
3.5, Assignee = "Janet Leverling", RankId = 6, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title = "Task -
29030", Status = "Testing", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Critical", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title = "Task -
29031", Status = "Testing", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title = "Task -
29032", Status = "Testing", Summary = "Check Login page validation.", Type =
"Story", Priority = "Release Breaker", Tags = "Testing", Estimate = 0.5,
Assignee = "Michael Suyama", RankId = 3 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title = "Task -
29033", Status = "Testing", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title = "Task -
29034", Status = "Testing", Summary = "Test editing functionality.", Type =
"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title = "Task -
29035", Status = "Testing", Summary = "Fix editing issues reported in
Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title = "Task -
29036", Status = "Testing", Summary = "Test editing feature in the IE
browser.", Type = "Story", Priority = "Normal", Tags = "Testing", Estimate =
5.5, Assignee = "Janet Leverling", RankId = 10 });
return TaskDetails;
}
}

```

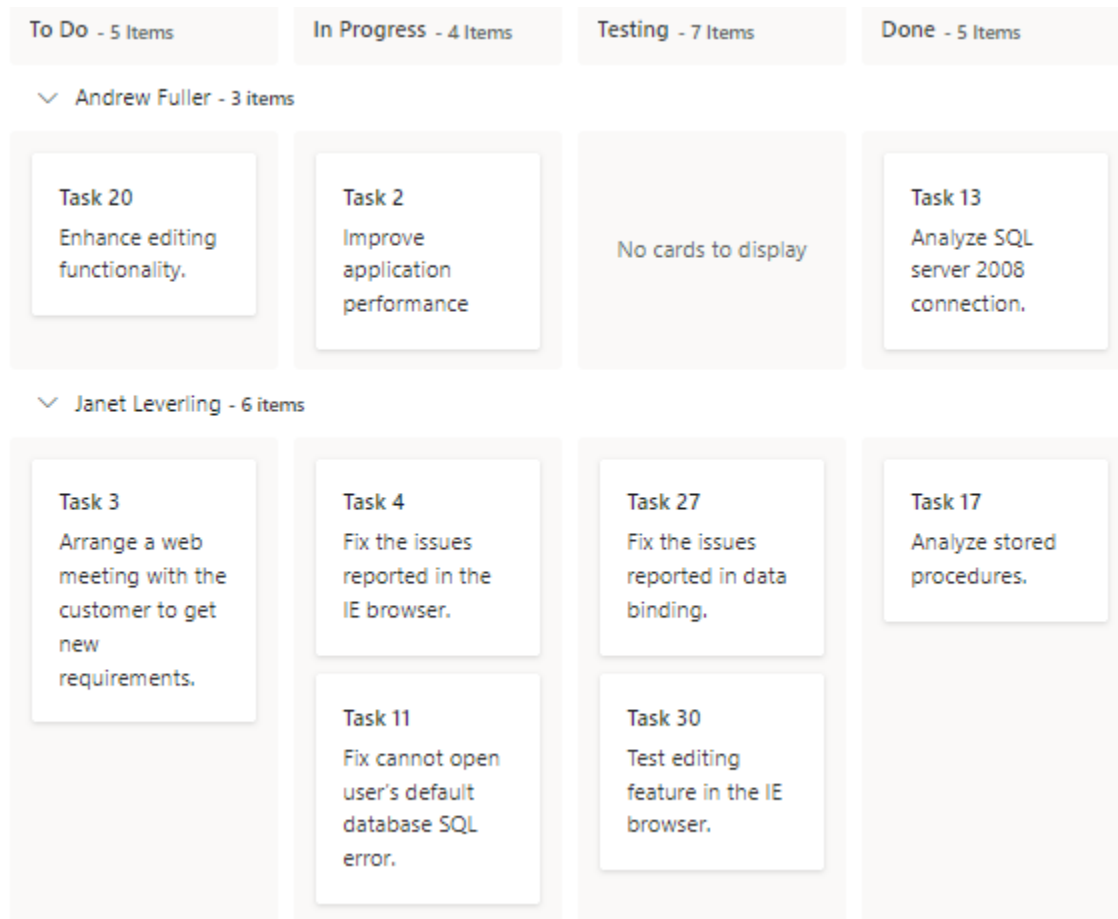


Enable swimlane

Swimlane can be enabled by mapping the tags [SwimlaneSettings.KeyField](#) to appropriate column name in dataSource. This enables the grouping of the cards based on the mapped column values.

CSHTML

```
@model List<KanbanSample.Controllers.KanbanDataModels>
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)Model).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee");
}).Render()
```



Note: [View Sample in GitHub.](#)

See also

- [Real time example using Kanban](#)

Columns in Kanban Board

The **Kanban** columns represent the each stage of the process. The column definitions are used as the **DataSource** schema in the Kanban. The Kanban operations such as drag-and-drop, swimlane, and toggle columns are performed based on column definitions.

Single-key mapping

Kanban columns are categorized by mapping the **Key** from the datasource using the **KeyField** property. The corresponding **Value** in the datasource is mapped inside the columns **KeyField**. Based on this categorization, Kanban columns are split on this board.

Note: The **KeyField** property is mandatory to render the columns in the Kanban board.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obj>
ct>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    }
}
```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",

```

```

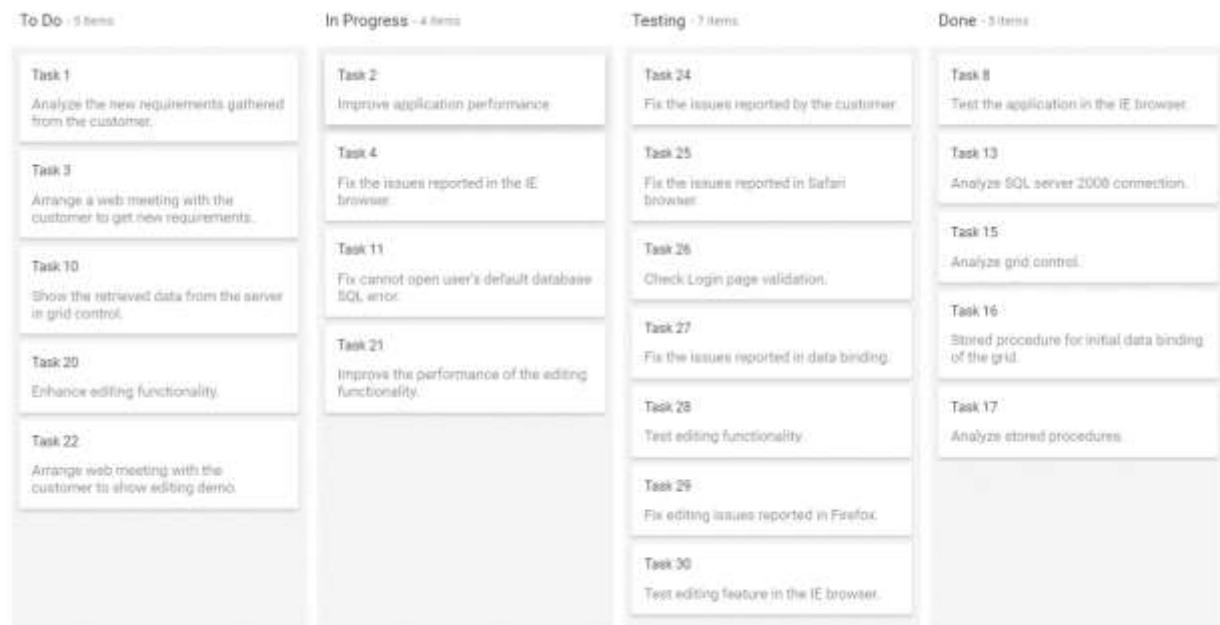
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.

**Multi-key mapping**

Kanban board allows to render a single column by mapping multiple keys using **KeyField** property. In below sample, specified the multiple keys(Open, Validate) to a single column.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open, Validate").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS


```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =

```



```

"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

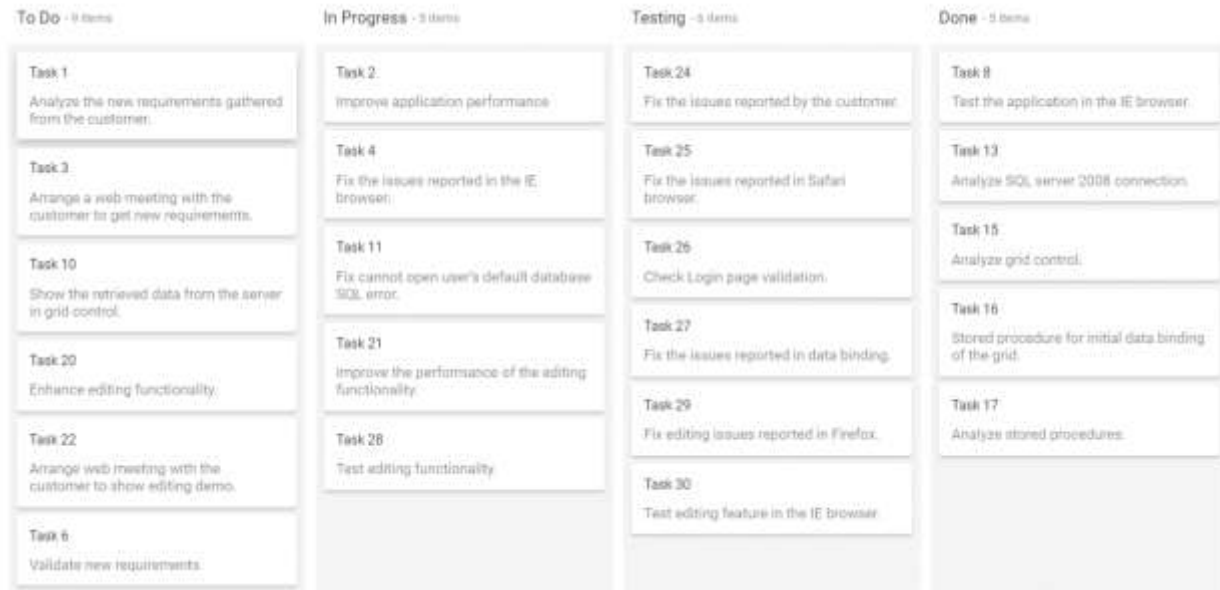
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Header text

You can provide the column header text of Kanban columns using the `HeaderText` property. If you have not specified any header text, it will render the header without any text.

Header template

You can customize the column header with any HTML or CSS element using the `Template` property as shown in the following code.

You can get the following columns data when using header template.

- `keyField`
- `headerText`
- `minCount`
- `maxCount`
- `allowToggle`
- `isExpanded`
- `showItemCount`
- `count`

CSHTML

```
<div>
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To
Do").KeyField("Open").Template("#headerTemplate").Add();
    col.HeaderText("In
Progress").KeyField("InProgress").Template("#headerTemplate").Add();
    col.HeaderText("In
Review").KeyField("Review").Template("#headerTemplate").Add();
    col.HeaderText("Done").KeyField("Close").Template("#headerTemplate").Add();
}).CardSettings(card => {
```

```

        card.ContentField("Summary").HeaderField("Id");
    }).Render()
</div>
<script id="headerTemplate" type="text/x-jsrender">
    <div class="header-template-wrap">
        <div class="header-icon e-icons ${keyField}"></div>
        <div class="header-text">${headerText}</div>
    </div>
</script>
<style type="text/css">
    .e-kanban .header-template-wrap {
        display: inline-flex;
        font-size: 15px;
        font-weight: 400;
    }
    .e-kanban .header-template-wrap .header-icon {
        font-family: 'Kanban priority icons';
        margin-top: 3px;
        width: 10%;
    }
    .e-kanban .header-template-wrap .header-text {
        margin-left: 15px;
    }
    .e-kanban.e-rtl .header-template-wrap .header-text {
        margin-right: 15px;
    }
    .e-kanban .card-header {
        padding-left: 12px;
    }
    .e-kanban .Open::before {
        content: '\e700';
        color: #0251cc;
        font-size: 16px;
    }
    .e-kanban .InProgress::before {
        content: '\e703';
        color: #ea9713;
        font-size: 16px;
    }
    .e-kanban .Review::before {
        content: '\e701';
        color: #8e4399;
        font-size: 16px;
    }
    .e-kanban .Close::before {
        content: '\e702';
        color: #63ba3c;
        font-size: 16px;
    }
    @@font-face {
        font-family: 'Kanban priority icons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMjltSfUAAAEoAAAAVmNtYXDnE+dkAAABlAAADxnbHlmg4w
eAgAAAdwAAAhQaGVhZBfH57sAAADQAAAAANmhoZWEIVQQGAAAArAAAACRobXR4FAAAAAAAYAAAAA
UbG9jYQNeBi4AAAHQAAAADG1heHABGAFgAAABCAAAACBuYW1lH65UOQAACiwAAALNcG9zdFsyKlE
AAAz8AAAAUgABAAAEAAAAAFwEAAAAAAD+AAABAAAAAAAAAAAAAAAAABQABAAAAAQAA7pb8lF8
PPPUACwQAAAAAANpY0WMAAAAAA2ljRYwAAAAAD+AP4AAAACAACAAAAAAAAAAAAEAAAAFAVQACQAAAAA

```

```

AAGAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAAACAaaaaAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAGADAAQAAAAAMwCBgKSBCgABAAAAAAD+AP4ACEAQwBlAKkAAAEfBw8HIS8HPwc
lHwcPByEvBz8HJR8HDwchLwc/BycRHw8hPw8RLw8hDw4CXgcGBQUEAwEBAQEDBAUFBgf+hgYGBQU
EAwEBAQEDBAUFBgYCOAYGBQUEAwEBAQEDBAUFBg9yAYGBQUEAwEBAQEDBAUFBgYCOAYGBQUEAwE
BAQEDBAUFBg9yAYGBQUEAwEBAQEDBAUFBgbcAQIDBQUHCAkKCgSMDQ00DQLgDQ4NDQwLCgoJCAc
FBQMCAQECaWUFBwgJCgoLDA0NDg39IA00DQ0MCwoKCQgHBQUdAgFDAQEDBAUFBgYHBgUFBAMBAQE
BAwQFBQYHBgYFBQDDAQG9AQEDBAUFBgCGBgUFBAMBAQEBAwQFBQYGBwYFBQDDAQG9AQEDBAUFBgY
HBgUFBAMBAQEBAwQFBQYHBgYFBQDDAQGz/SANDg0NDASKCgkIBwUFAwIBAQIDBQUHCAkKCgSMDQ0
ODQLgDQ4NDQwLCgoJCAcFBQMCAQECaWUFBwgJCgoLDA0NDgAABAAAAAAD+AP4AD8AggDUARgAAAE
fBw8PLw41Pw8fBicPDx8PMz8OLxAHNzMFehUPESsBLxA9AT8UJREFdYe/DxEvDyEPDgJlCacGBgQ
CAGeBAGMEBQcHCAkJCwsMDAwNDgWNDASLCgkICAYFAwMBAQMDBQUHBwgJCQoLCwwMDA4MDAwLCgq
EDg8PDw4PDw8VFBQUEXMTehUWFhYXFxgYehMSERISEREUEBEREBESERkZGRgXFxcXEA8QEBARERE
WFxYVfHUFhIefASXGBkYGRkYGSATEXISEhIRBQMBAGICHBkaGhscGx0UEXMTExMTExoUFRQVFBu
VHBoaGhkYGRkeAgIDGBQVfHXYFxcREREQEREQE8ODv4aAQIDBQUHCAkKCgSMDQ00DQLgDQ4NDQw
LCgoJCAcFBQMCAQECaWUFBwgJCgoLDA0NDg39IA00DQ0MCwoKCQgHBQUdAgJXCQoKCwsMDAwNDAw
MCgSJCQgHBgUEAwIBAQIDBQUHCAkKCgSMCw0MDQwLDAoLCQkJBwcGBQQCAGeBAGMEBQYIWQMEBQY
GBwgJDg4PERETEXUYfXUTEhAPDgkIBwUFAwEBAgIEBQYHCA0QEBMUfHcaEREQDw8NDQ0PDQsJCAy
EAwEBMAIEBgJDA4PFg8PERESFBQHBwYGBgUEIBsZFhUTERAJCAyGBAMCAGQFBggJChAREhUWGBBo
eCAUFBAyHGxcVFBMREQ8KCQgHBgYEBAMCAYT9IA00DQ0MCwoKCQgHBQUdAgEBAgMFBQcICQoKCww
NDQ4NAuANDg0NDASKCgkIBwUFAwIBAQIDBQUHCAkKCgSMDQ00AAIAAAAAA/gD+AArAG8AAAEfAhU
PAwEPay8INT8GMx8DAT8DHwILER8PIT8PES8PIQ8OavMEAwIBAQME/r8FBQYGBgYFBXkEAWEBAGM
EBQUGBgYGBgViASoFBgYGBgYF/ROBAgMFBQcICQoKCwwNDQ4NAuANDg0NDASKCgkIBwUFAwIBAQI
DBQUHCAkKCgSMDQ00Df0gDQ4NDQwLCgoJCAcFBQMCArQFBgYGBgYFBf7FBAMBAQEBAwR2BQUGBgY
GBgUEAwEBAGMEYAE1BAMBAQEBA7j9IA00DQ0MCwoKCQgHBQUdAgEBAgMFBQcICQoKCwwNDQ4NAuA
NDg0NDASKCgkIBwUFAwIBAQIDBQUHCAkKCgSMDQ00AAAJAAAAAP4A/gAIQBDAGUAhwCpAMsA7QE
PAVMAAAEVDwcvBzU/Bx8GNx8EDwYrAS8GPQE/BTSBHwEFHwMPBysBLwU9AT8GOWEfASUfBw8HIy8
HPwchHwcPByMvBz8HJR8DDwcrAS8FPQE/BjsBHwEFHwMdAQ8FKwEvBz8GOWEfASUVDwcvBzU/Bx8
GJREFdYe/DxEvDyEPDgIgAQIDBAQGBgYGBgYEBAMCAQECaWQEBgYGBgYGBAQDAopiBAMCAQECaWQ
FBQYGBgYFBWIEAwICaWQFBQYGBgYF/t8EAWIBAQIDBGIQFBQYGBgYFBQDDAgIDBGIQFBQYGBgYFAdw
HBgUFBAMBAQEBAWQFBQYHigYGBgQEAWIBAQIDBAQGBgb+YAYGBgQEAWIBAQIDBAQGBgaKBwYFBQD
DAQEBAQMEBQUGBwJlBAMCAQECaWRiBQUGBgYGBQUEAwICaWRiBQUGBgYGBf4bYgQDAgIDBAUFBgY
GBgUFYgQDAgEBAgMEBQUGBgYGBQEEAQIDBAQGBgYGBgYEBAMCAQECaWQEBgYGBgYGBAQDAv3pAQI
DBQUHCAkKCgSMDQ00DQLgDQ4NDQwLCgoJCAcFBQMCAQECaWUFBwgJCgoLDA0NDg39IA00DQ0MCwo
KCQgHBQUdAgEwigcGBQUEAwEBAQEDBAUFBgEKBgYGBAQDAgEBAgMEBAYGTWIFBQYGBgYFBQDDAgI
DBGIFBQYGBgYFBQDDAgIDBAUFBgYGBgUFYgQDAgIDBAUFBgYGBgUFYgQDAgIDmQECaWQEBgYGBgY
GBAQDAgEBAgMEBAYGBgYGBgQEAWIBAQIDBAQGBgYGBgYEBAMCAQECaWQEBgYGBgYGBAQDAgHrBQU
GBgYGBQViBAMCAGMEBQUGBgYGBQViBAMCAGMEYgUFBgYGBgUFBAMCAGMEYgUFBgYGBgUFBAMCAGN
LigYGBgQEAWIBAQIDBAQGBgaKBwYFBQDDAQEBAQMEBQUd/0gDQ4NDQwLCgoJCAcFBQMCAQECaWU
FBwgJCgoLDA0NDg0C4A00DQ0MCwoKCQgHBQUdAgEBAgMFBQcICQoKCwwNDQ4AAAAAEgDeAAAAAA
AAAAAQAAAAEAAAAAAAEAFQABAAEAAAAAAAIABwAAAEAAAAAAAMAFQAdAAEAAAAAAQAfQAYAAE
AAAAAAUACwBHAAEAAAAAAAYAFQBSAAEAAAAAAALABNAEEAAAAAAASAEgCTAAMAAQQJAAAAAGC
laAMAAQQJAEEAKgCnaAMAAQQJAaiADgDRAAMAAQQJAAMAKgDfAAMAAQQJAaQAKgEJAAMAAQQJAAU
AFgEzAAMAAQQJAAYAKgFJAAMAAQQJAAoAWAFzAAMAAQQJAAsAJAHLIEthbmJhbiBwcmllvcml0eSB
pY29uc1JlZ3VsYXJlYW5iYW4gcHJpb3JpdHkgaWNvbnNlYW5iYW4gcHJpb3JpdHkgaWNvbnNWZXJ
zaW9uIDEuMETHbmJhbiBwcmllvcml0eSBpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAASwBhAG4AYgBhAG4AIABwAHIAaQB
vAHIAQB0AHkAIAIBpAGMabwBuAHMAUgBlAGcAdQBsAGEAcgBLAGEAbgBiAGEAbgAgAHAacgBpAG8
AcgBpAQeQAgAGkAYwBvAG4AcwBLAGEAbgBiAGEAbgAgAHAacgBpAG8AcgBpAQeQAgAGkAYwB
vAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAASwBhAG4AYgBhAG4AIABwAHIAaQBvAHIAaQB0AHk
AIAIBpAGMabwBuAHMARgBvAG4AdAAgAGcAZQBwAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB
5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdAB1AGQAaQBvAHcAdwB3AC4AcwB5AG4
AYwBmAHUAcwBpAG8AbgAuAGMabwBtAAAAAIAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ABQECAQMBBAEFAQYACFRvZG9saXN0BlJldmllldwldB21wbGV0ZWQIUHJvZ3Jlc3MAAAAA)
format('trueType');
font-weight: normal;

```

```

        font-style: normal;
    }
    [class^="sf-icon-"],
    [class*=" sf-icon-"] {
        font-family: 'Kanban priority icons' !important;
        speak: none;
        font-size: 55px;
        font-style: normal;
        font-weight: normal;
        font-variant: normal;
        text-transform: none;
        line-height: 1;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
    }
</style>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",

```

```

Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the

```

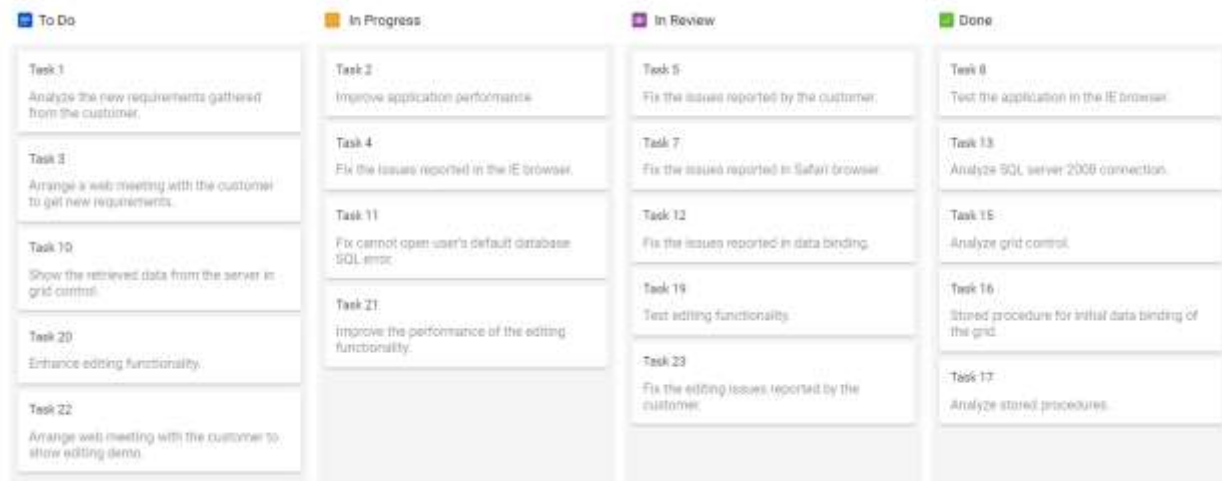


```
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.



Toggle columns

Kanban allows to expand or collapse its columns using `AllowToggle` inside the `Columns` property. When enable the property, it will render the expand or collapse icon to the column header.

Note: By default, collapsed column width is set to 50px.

CSHTML

```
<div>
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").AllowToggle(true).Add();
    col.HeaderText("In
Progress").KeyField("InProgress").AllowToggle(true).Add();

    col.HeaderText("Testing").KeyField("Testing").AllowToggle(true).Add();
    col.HeaderText("Done").KeyField("Close").AllowToggle(true).Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
});
</div>
```

```

    }).Render()
</div>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",

```

```

Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });

return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
    }
}

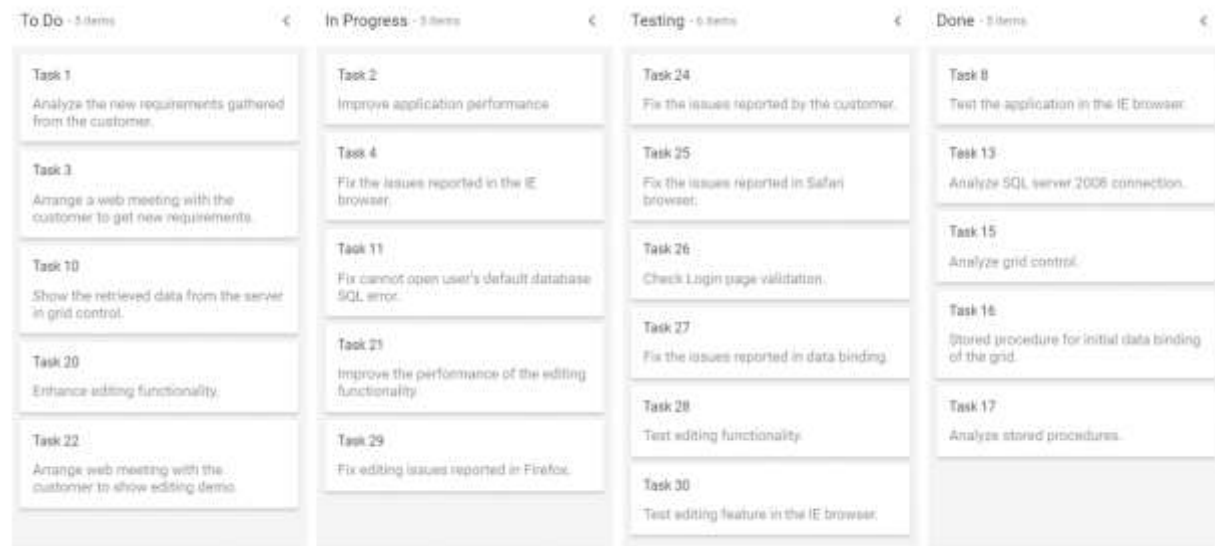
```

```

        return View();
    }
}

```

Output be like the below.



Initially collapsed column

By default, all columns are on expanded state when loading the Kanban board initially. But, you can render the columns with collapsed state using the `IsExpanded` property.

Note: The `IsExpanded` property only works when enabling the `AllowToggle` property on particular column.

In the following example, the To Do column is collapsed on initialization of Kanban board.

CSHTML

```

<div>
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").AllowToggle(true).IsExpanded(false).Add();
    col.HeaderText("In Progress").KeyField("InProgress").AllowToggle(true).Add();

    col.HeaderText("Testing").KeyField("Testing").AllowToggle(true).IsExpanded(false).Add();
    col.HeaderText("Done").KeyField("Close").AllowToggle(true).Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
</div>

```

DATASOURCE.CS

```

public class KanbanDataModels

```

```

{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =

```

```

"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```



```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

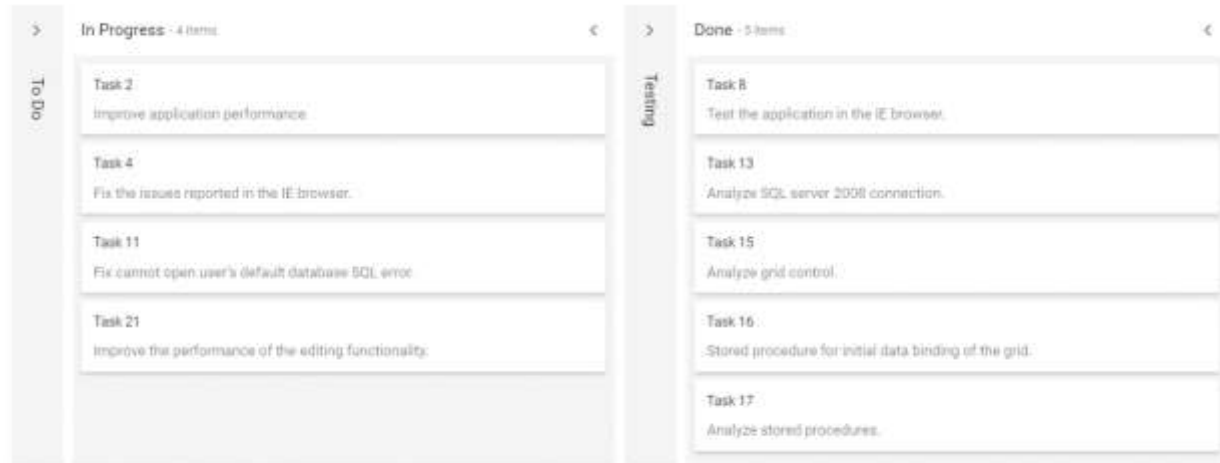
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Stacked headers

Stacked headers are the additional headers to column header that will group the similar columns.

Define the grouping of columns **Key** value to the **KeyFields** property and provide the custom header text name to grouped columns using the **Text** property, which is placed inside the **StackedHeaders** property.

In the following code, the kanban columns 'InProgress, Review' are grouped under 'Development phase' category.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("Open").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("In Review").KeyField("Review").Add();
    col.HeaderText("Completed").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).StackedHeaders(stackedHeaders => {
    stackedHeaders.Text("To Do").KeyFields("Open").Add();
    stackedHeaders.Text("Development Phase").KeyFields("InProgress, Review").Add();
    stackedHeaders.Text("Done").KeyFields("Close").Add(); }).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
}
```

```

public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =

```

```

"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

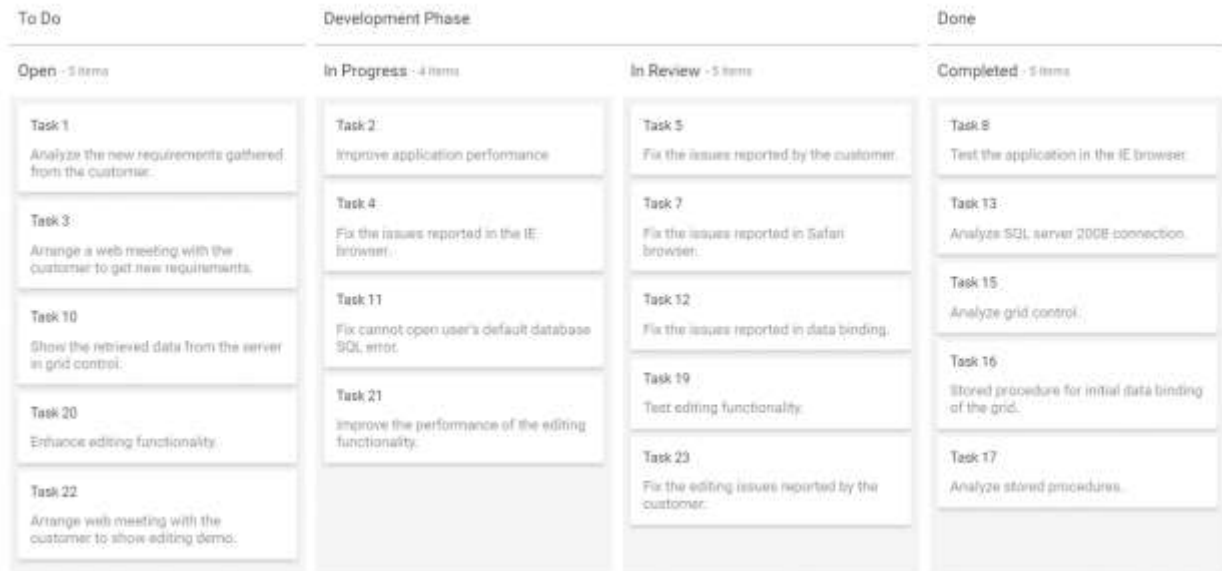
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Cards in ASP.NET MVC Kanban component

The cards are main elements in Kanban board, which represent the task information with header and content. The header and content of a card is fetched from the corresponding mapping fields. The card layout can be customized with template also.

Drag-and-drop

Transit or change the card position using the drag-and-drop functionality. By default, the `AllowDragAndDrop` property is enabled on the Kanban board, which is used to change the card position by column-to-column or within the column.

Added dotted border on Kanban cells except the dragged clone cells when dragging, which indicates the possible ways for dropping the cards into the cells.

Header

The card header is achieved by mapping the `HeaderField` property, which is placed inside the `CardSettings` property. By default, the `ShowHeader` property enabled by Kanban board that is used to show the header at the top of the card.

Note: The `HeaderField` property of `CardSettings` is mandatory to render the cards in the Kanban board. It acts as a unique field that is used to avoid the duplication of card data. You can not change the `HeaderField` of mapped data value using the `updateCard` public method or server-side update of data.

In the following demo, the `ShowHeader` property is disabled on Kanban board.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
```

```
card.ContentField("Summary").HeaderField("Id").ShowHeader(false);
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    }
}
```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",

```



```

Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });

return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
    }
}

```

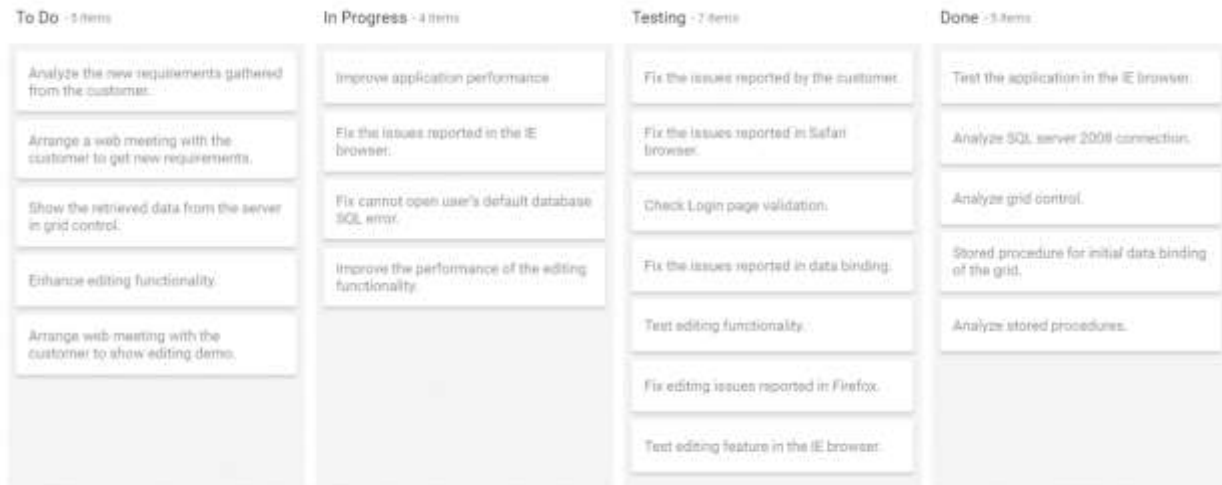


```

        return View();
    }
}

```

Output be like the below.



Content

The card's content is fetched from data source using the **ContentField** property, which is placed inside the **CardSettings** property. If the **ContentField** property is not used, card is rendered with empty content.

Template

You can customize the default card layout using template as per your application needs. This can be achieved by template of the **CardSettings** property.

CSSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Category").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("Menu").KeyField("Menu").Add();
    col.HeaderText("Order").KeyField("Order").Add();
    col.HeaderText("Ready to Serve").KeyField("Ready to
Serve").Add();
    col.HeaderText("Delivered").KeyField("Delivered, Served").Add();
}).CardSettings(card => {
    card.HeaderField("Id").Template("#cardTemplate");
}).Render()
<script id="cardTemplate" type="text/x-jsrender">
<div class='card-template'>
    <div class='card-template-wrap'>
        <table class='card-template-wrap'>
            <colgroup>
                <col style="width:35px">
                <col style="width:calc(100% - 45px)">
            </colgroup>

```

```
<tbody>
  <tr>
    <td class='e-image'>
      
    </td>
    <td class='e-title'>
      <div class="e-card-stacked">
        <div class='e-card-header'>
          <div class='e-card-header-caption'>
            <div class='e-card-header-title
e-tooltip-text'>${Title}</div>
          </div>
        </div>
        <div class="e-card-content" style="line-
height:2.75em">
          <table class='card-template-wrap'>
            <tbody>
              <tr class='e-tooltip-text'>
                ${if(Category == "Menu"
|| Category=="Order" || Category=="Ready to Serve") }
                <td colspan="2">
                  <div class='e-
description'>
                    ${if(Category
=="Menu")}
                    ${Description}
                    ${else}
                    ${OrderID}
                    ${/if}
                  </div>
                </td>
                ${else}
                <td><div class='e-
description'>${OrderID}</div></td>
                <td><span class='e-icons
e-done'></span></td>
                ${/if}
              </tr>
              <tr>
                ${if(Category != "Menu")}
                ${if(Category
=="Order")}
                <td><div class='e-
preparingText e-tooltip-text'>Preparing</div></td>
                <td class='e-prepare'>
                  <div class='e-time
e-tooltip-text'>
                    <div class='e-
icons e-clock'></div><div class='e-mins'>15 mins</div>
                  </div>
                </td>
                ${/if}
                ${if(Category == "Ready
to Serve")}
                <td><div class='e-
readyText e-tooltip-text'>Ready to Serve</div></td>
```

```

<td class='e-prepare'>
    <div class='e-time
e-tooltip-text'>
        <div class='e-
icons e-clock'></div><div class='e-mins'>5 mins</div>
        </div>
    </td>
    ${if}
    ${if(Category
=="Delivered" || Category=="Served") }
    <td><div class='e-
deliveredText e-tooltip-text'>Delivered</div></td>
    ${if}
    ${else}
    <td><div class='e-size
e-tooltip-text'>${Size}</div></td>
    <td><div class='e-price
e-tooltip-text'>${Price}</div></td>
    ${if}
    </tr>
</tbody>
</table>
</div>
</div>
</td>
</tr>
</tbody>
</table>
</div>
</div>
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string OrderID { get; set; }
    public string Title { get; set; }
    public string Type { get; set; }
    public string Size { get; set; }
    public string Status { get; set; }
    public string Category { get; set; }
    public string Price { get; set; }
    public string Description { get; set; }
    public string Tags { get; set; }
    public string ImageURL { get; set; }
    public List<KanbanDataModels> KanbanPizzaData()
    {
        List<KanbanDataModels> PizzaData = new List<KanbanDataModels>();
        PizzaData.Add(new KanbanDataModels { Id = "1", OrderID = "Order
ID - #16365", Title = "Mexican Green Wave", Type = "Vegetarian", Size =
"Small", Category = "Order", Description = "Stromboli sandwich with chili
sauce.", Tags = "Onions Pepper Cheese", ImageURL = "menu_01" });
        PizzaData.Add(new KanbanDataModels { Id = "2", OrderID = "Order
ID - #16366", Title = "Milan Veg Fantasy", Type = "Vegetarian", Size =

```

```

"Medium", Category = "Order", Description = "Zucchini wrapped in spicy
grilled seasoning along with tomato and jalapeno.", Tags = "Onions Pepper
Tomato Zucchini", ImageURL = "menu_01" });
    PizzaData.Add(new KanbanDataModels { Id = "3", OrderID = "Order
ID - #16367", Title = "Peppy Paneer", Type = "Vegetarian", Size = "Large",
Category = "Ready to Serve", Description = "It's made using toppings of
tomato mozzarella cheese and fresh basil which represent the red white and
green of the Italian flag.", Tags = "Onions Pepper Cheese", ImageURL =
"menu_02" });
    PizzaData.Add(new KanbanDataModels { Id = "4", OrderID = "Order
ID - #16368", Title = "Margherita", Type = "Vegetarian", Size = "Small",
Category = "Menu", Description = "Lebanese Pizza topped with tomato sauce.",
Tags = "Onions Pepper Cheese", ImageURL = "menu_03", Price = "$4.79" });
    PizzaData.Add(new KanbanDataModels { Id = "5", OrderID = "Order
ID - #16369", Title = "Farm House", Type = "Vegetarian", Size = "Small",
Category = "Delivered", Description = "Stromboli sandwich with chili
sauce.", Tags = "Onions Pepper Cheese", ImageURL = "menu_04" });
    PizzaData.Add(new KanbanDataModels { Id = "6", OrderID = "Order
ID - #16370", Title = "BBQ Chicken", Type = "Non-Vegetarian", Size =
"Medium", Category = "Ready to Serve", Description = "BBQ Chicken with chili
sauce.", Tags = "Onions Pepper Chicken BBQ", ImageURL = "menu_05", Price =
"$4.79" });
    PizzaData.Add(new KanbanDataModels { Id = "7", OrderID = "Order
ID - #16371", Title = "Tandoori Chicken", Type = "Non-Vegetarian", Size =
"Large", Category = "Ready to Serve", Description = "Tandoori Chicken with
chili sauce.", Tags = "Onions Tandoori Pepper Chicken", ImageURL = "menu_06"
});
    PizzaData.Add(new KanbanDataModels { Id = "8", OrderID = "Order
ID - #16372", Title = "BBQ Prawn", Type = "Non-Vegetarian", Size = "Large",
Category = "Order", Description = "BBQ Prawn with chili sauce.", Tags =
"Onions BBQ Pepper Prawn", ImageURL = "menu_07" });
    PizzaData.Add(new KanbanDataModels { Id = "9", OrderID = "Order
ID - #16373", Title = "Italian Chicken", Type = "Non-Vegetarian", Size =
"Medium", Category = "Menu", Description = "Italian Chicken with White
sauce.", Tags = "Onions Pepper Italian Chicken", ImageURL = "menu_08", Price
= "$11.99" });
    PizzaData.Add(new KanbanDataModels { Id = "10", OrderID = "Order
ID - #16374", Title = "Garlic Prawn", Type = "Non-Vegetarian", Size =
"Small", Category = "Delivered", Description = "Prawn with chili sauce.",
Tags = "Onions Garlic Pepper Prawn", ImageURL = "menu_13" });
    PizzaData.Add(new KanbanDataModels { Id = "11", OrderID = "Order
ID - #16375", Title = "Double Cheese Margherita", Type = "Vegetarian", Size
= "Medium", Category = "Delivered", Description = "Margherita with chili
sauce and double Cheese.", Tags = "Onions, Pepper, Cheese", ImageURL =
"menu_10", Price = "$11.99" });
    PizzaData.Add(new KanbanDataModels { Id = "12", OrderID = "Order
ID - #16376", Title = "Veggie Delight", Type = "Vegetarian", Size = "Large",
Category = "Menu", Description = "Veggie Delight with chili sauce and extra
toppings.", Tags = "Onions, Capsicum, Pepper, Cheese", ImageURL = "menu_11",
Price = "$14.99" });
    PizzaData.Add(new KanbanDataModels { Id = "13", OrderID = "Order
ID - #16377", Title = "Paneer Tikka", Type = "Vegetarian", Size = "Large",
Category = "Order", Description = "Paneer Tikka with chili sauce.", Tags =
"Onions, Paneer, Pepper, Tikka", ImageURL = "menu_12", Price = "$14.99" });
    PizzaData.Add(new KanbanDataModels { Id = "14", OrderID = "Order
ID - #16378", Title = "Chicken Tikka", Type = "Non-Vegetarian", Size =
"Medium", Category = "Ready to Serve", Description = "Chicken Tikka with

```

```

White sauce.", Tags = "Onions, Pepper, Chicken, Tikka, Boneless", ImageURL =
"menu_14", Price = "$11.99" });
    PizzaData.Add(new KanbanDataModels { Id = "15", OrderID = "Order
ID - #16379", Title = "Kadai Chicken", Type = "Non-Vegetarian", Size =
"Small", Category = "Menu", Description = "Kadai Chicken with chili sauce.",
Tags = "Onions, Pepper, Chicken", ImageURL = "menu_15", Price = "$4.79" });
    PizzaData.Add(new KanbanDataModels { Id = "16", OrderID = "Order
ID - #16380", Title = "Hot Vege", Type = "Vegetarian", Size = "Medium",
Category = "Delivered", Description = "Capsicum with chili sauce and Double
Cheese with extra toppings.", Tags = "Onions, Pepper, Capsicum, Cheese",
ImageURL = "menu_16", Price = "$11.99" });
    PizzaData.Add(new KanbanDataModels { Id = "17", OrderID = "Order
ID - #16381", Title = "Kadai Paneer", Type = "Vegetarian", Size = "Large",
Category = "Menu", Description = "Kadai Paneer with chili sauce and extra
toppings.", Tags = "Onions, Capsicum, Pepper, Paneer", ImageURL = "menu_17",
Price = "$14.99" });
    PizzaData.Add(new KanbanDataModels { Id = "18", OrderID = "Order
ID - #16382", Title = "Tomato Pizza", Type = "Vegetarian", Size = "Large",
Category = "Served", Description = "Tomato Pizza with chili sauce and extra
toppings.", Tags = "Onions, Tomato, Pepper, Capsicum", ImageURL = "menu_18",
Price = "$14.99" });
    return PizzaData;
}
}

```

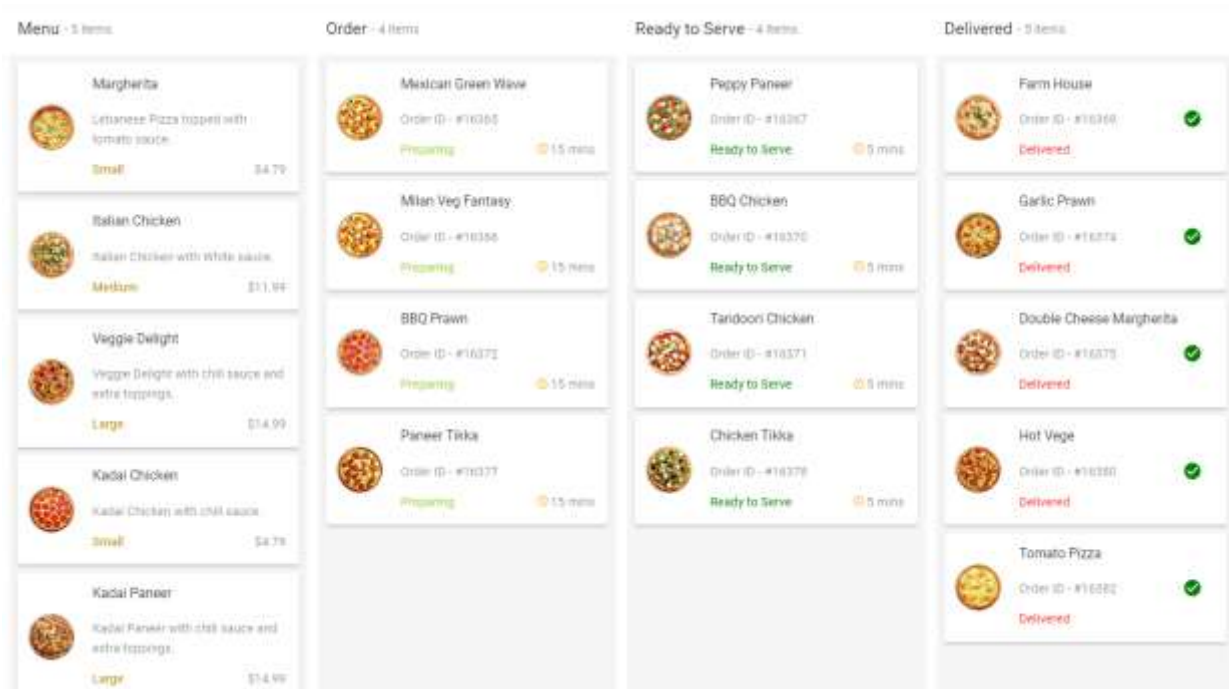
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanPizzaData();
        return View();
    }
}

```

Output be like the below.



Selection

Kanban board allows to select single and multiple selection of cards when mouse or keyboard interactions using **SelectionType** property. The property contains following types.

- **None:** No cards are allowed to select from Kanban board.
- **Single:** Only one card allowed to select at a time in the Kanban board.
- **Multiple:** Multiple cards are allowed to select in a board.

Multiple Selection

Select the multiple cards randomly using Ctrl + mouse click and select the multiple cards continuously using Shift + mouse click action on Kanban board. Set **Multiple** in **SelectionType** to enable the multiple selection in a board.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id").SelectionType(SelectionType.Multiple);
}).Render()
```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =

```

```

"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```



```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

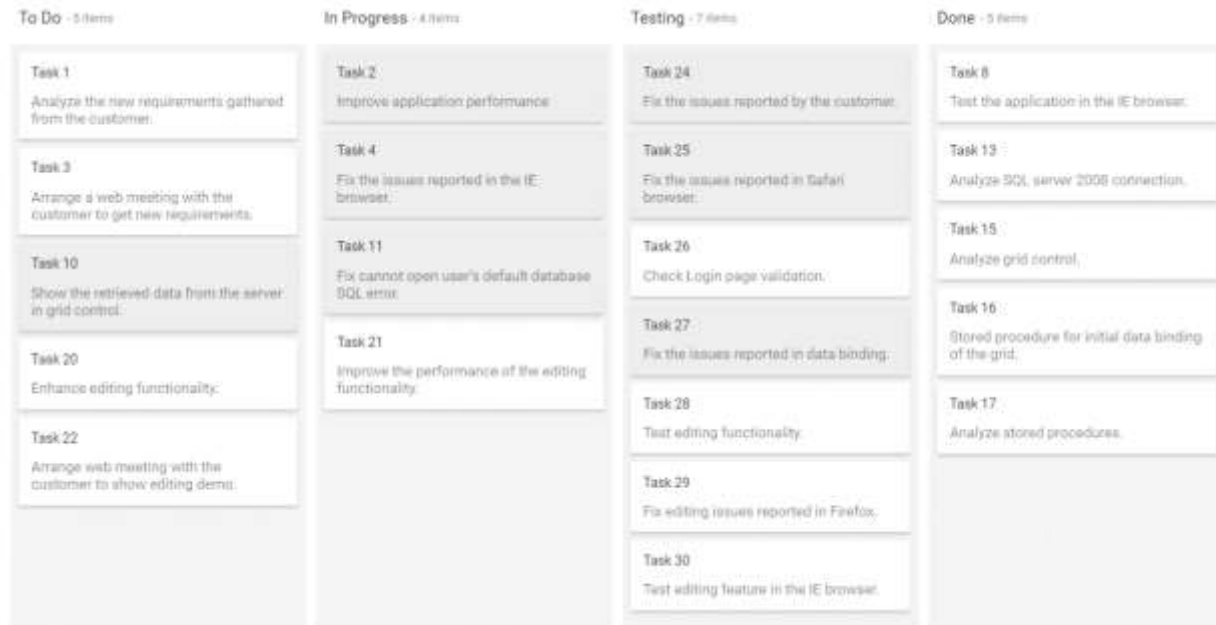
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Data binding in ASP.NET MVC Kanban component

The Kanban uses **DataManager**, which supports both RESTful data service binding and list binding. The **DataSource** property of Kanban can be assigned either with the instance of **DataManager** or List, as it supports the following two data binding methods:

- Local data
- Remote data

Local data

To bind local list data to the Kanban, you can simply assign a list to the **DataSource** property. The list can also be provided as an instance of **DataManager** and assigned to the Kanban **DataSource** property.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
}
```

```

public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from

```

```

the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =

```

```

"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

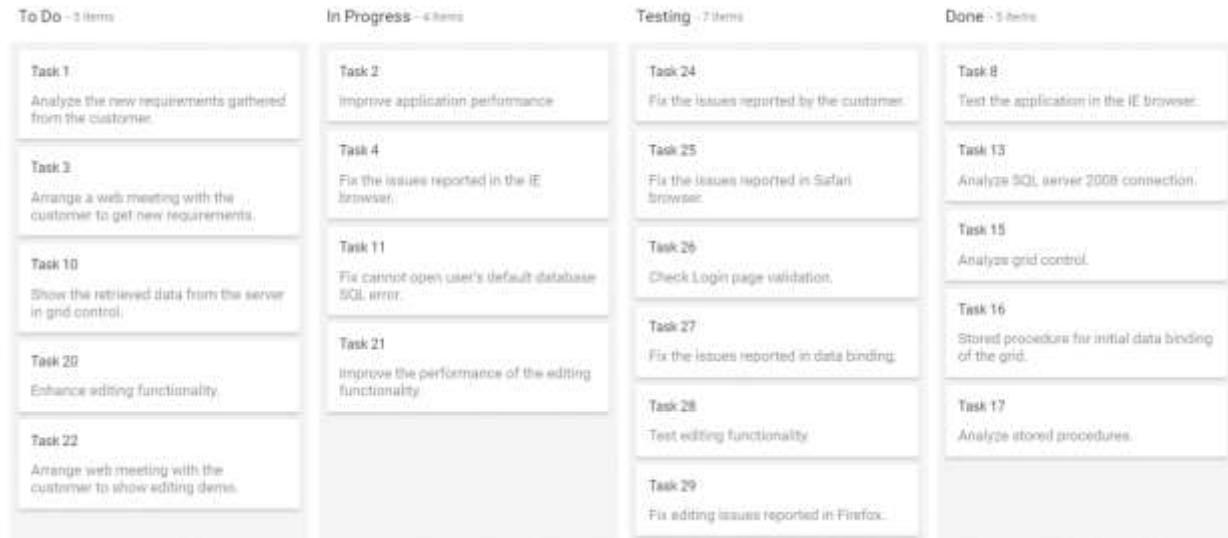
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Note: By default, **DataManager** uses **JsonAdaptor** for binding local data.

Remote data

To bind remote data to kanban component, assign service data as an instance of **DataManager** to the **DataSource** property. To interact with remote data source, provide the endpoint **url**.

CSHTML

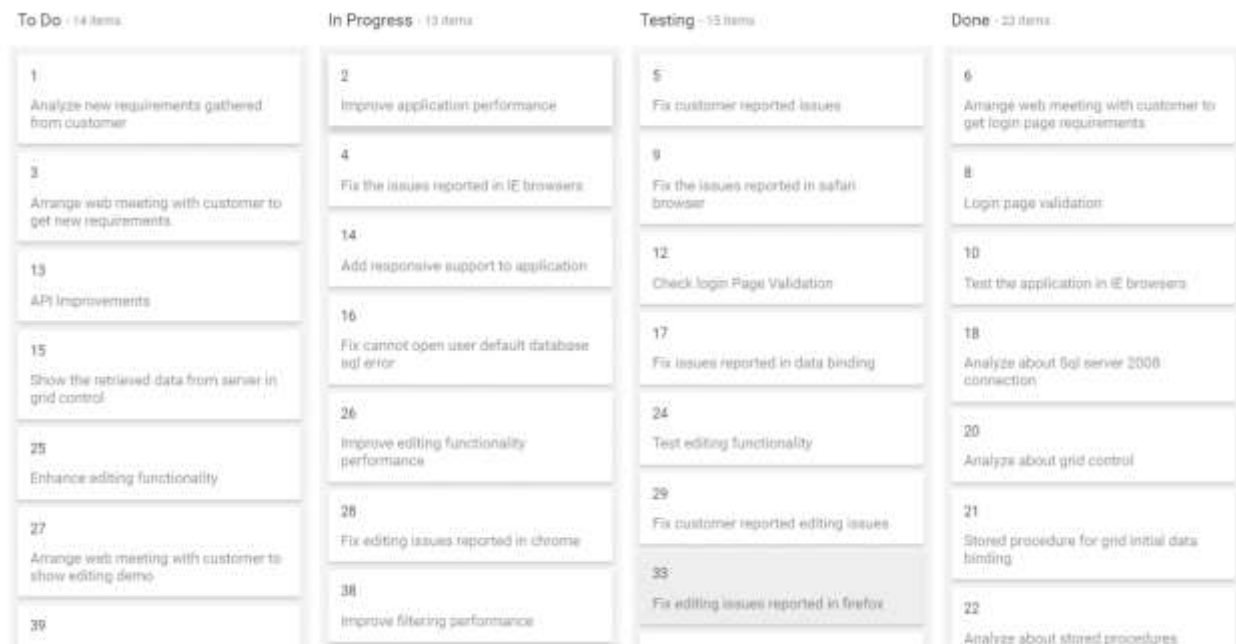
```
@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).DataSource(dataManger =>
{
    dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/Kanban").CrossDomain(true);
}).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DialogOpen("onDialogOpen").Render()

<script>
function onDialogOpen(args) {
    args.cancel = true;
}
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Output be like the below.



Note: By default, **DataManager** uses **ODataAdaptor** for remote data-binding.

OData services

[OData](#) is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).DataSource(dataManger => {

    dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/Kanban").CrossDomain(true).Adaptor("ODataAdaptor");
    }).Columns(col=> {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id");
    }).DialogOpen("onDialogOpen").Render()

<script>
    function onDialogOpen(args) {
        args.cancel = true;
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
```

```
{
    public ActionResult Index()
    {
        return View();
    }
}
```

OData v4 services

The ODataV4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [OData Documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).DataSource(dataManger => {

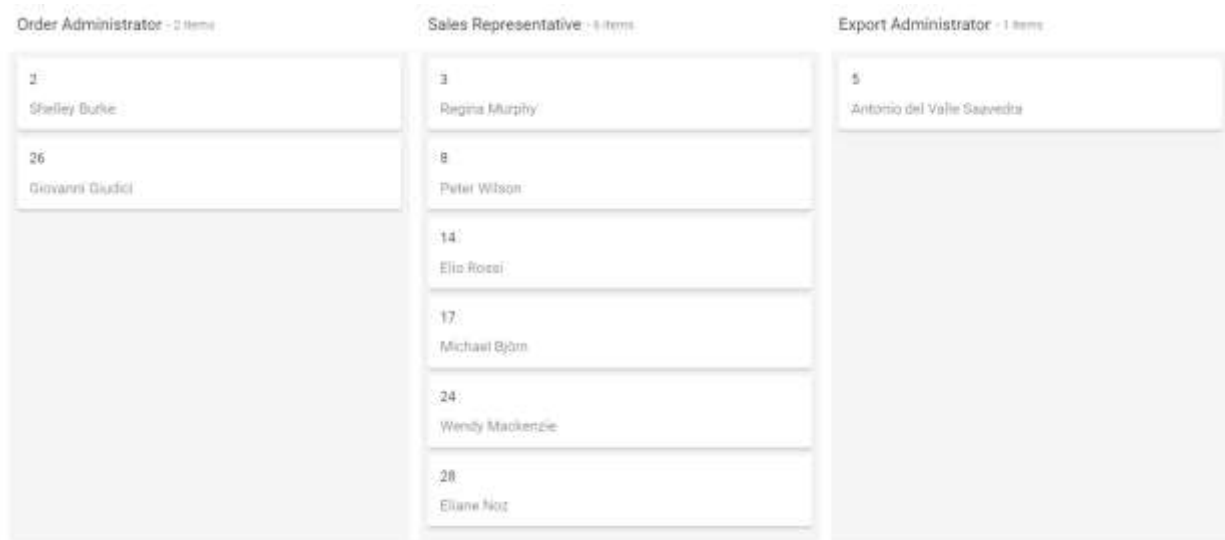
    dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/Kanban").CrossDomain(true).Adaptor("ODataAdaptor");
    }).Columns(col=> {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id");
    }).DialogOpen("onDialogOpen").Render()

<script>
    function onDialogOpen(args) {
        args.cancel = true;
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Output be like the below.



Web API

You can use **WebApiAdaptor** to bind kanban with Web API created using OData endpoint.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).DataSource(dataManger =>
{
    dataManger.Url("/api/Tasks").CrossDomain(true).Adaptor("WebApiAdaptor");
}).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DialogOpen("onDialogOpen").Render()
<script>
    function onDialogOpen(args) {
        args.cancel = true;
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Below server-side controller code to get the Kanban data.

`typescript

[HttpGet]

public object Get()

```
{
var data = OrdersDetails.GetAllRecords().ToList();
return data;
}
```

URL adaptor

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Kanban cards using the various adaptors available within the **DataManager**. Most preferably, we will be using **UrlAdaptor** for performing CRUD actions on Kanban.

The CRUD operation in Kanban can be mapped to server-side controller actions using the properties **InsertUrl**, **RemoveUrl**, **UpdateUrl**, and **CrudUrl**.

- **InsertUrl** – You can perform a single insertion operation on the server-side.
- **UpdateUrl** – You can update single data on the server-side.
- **RemoveUrl** – You can remove single data on the server-side.
- **CrudUrl** – You can perform bulk data operation on the server-side.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource(dataManger =>
{
dataManger.Url("/Home/DataSource").UpdateUrl("/Home/Update").InsertUrl("/Home/Insert").RemoveUrl("/Home/Delete").CrossDomain(true).Adaptor("UrlAdaptor");
}).Columns(col=> {
col.HeaderText("To Do").KeyField("Open").Add();
col.HeaderText("In Progress").KeyField("InProgress").Add();
col.HeaderText("Testing").KeyField("Testing").Add();
col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
card.ContentField("Summary").HeaderField("Id");
}).Render()
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

The server-side controller code to handle the CRUD operations are as follows.

```

`typescript
private NORTHWNDEntities db = new NORTHWNDEntities();
public ActionResult DataSource() {
    var DataSource = db.Tasks.ToList();
    return Json(DataSource, JsonRequestBehavior.AllowGet);
}
public ActionResult Insert(Params value) {
    //Insert card data into the database
    return Json(value, JsonRequestBehavior.AllowGet);
}
public ActionResult Update(Params value) {
    //Update card data into the database
    return Json(value, JsonRequestBehavior.AllowGet);
}
public void Delete(Params value) {
    //Delete card data from the database
}
public class Params {
    public int Id { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Assignee { get; set; }
}
`

```

Note: The `CrudUrl` is used to update the bulk data sent to the server-side. Multiple selections and `SortBy` as `Index` properties are used for `CrudUrl` properties to update the modified bulk data to the server-side.

Custom adaptor

It is possible to create your own custom adaptor by extending the built-in available adaptors. The following example demonstrates the custom adaptor usage and how to add a custom field `TaskId` for the cards by overriding the built-in response processing using the `ProcessResponse` method of the `ODataAdaptor`.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).Columns(col=> {

```

```

        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id");
    }).DialogOpen("onDialogOpen").Created("onCreate").Render()
</script>
function onDialogOpen(args) {
    args.cancel = true;
}
function onCreate(args) {
    class TaskIdAdaptor extends ej.data.ODataAdaptor {
        processResponse() {
            var i = 0;
            // calling base class processResponse function
            var original = super.processResponse.apply(this, arguments);
            original.forEach((item) => item['Id'] = 'Task - ' + ++i);
            return original;
        }
    }
    var kanban = document.querySelector('#kanban').ej2_instances[0];
    kanban.dataSource = new ej.data.DataManager({
        url: "https://ej2services.syncfusion.com/production/web-
services/api/Kanban",
        adaptor: new TaskIdAdaptor()
    });
}
</script>

```

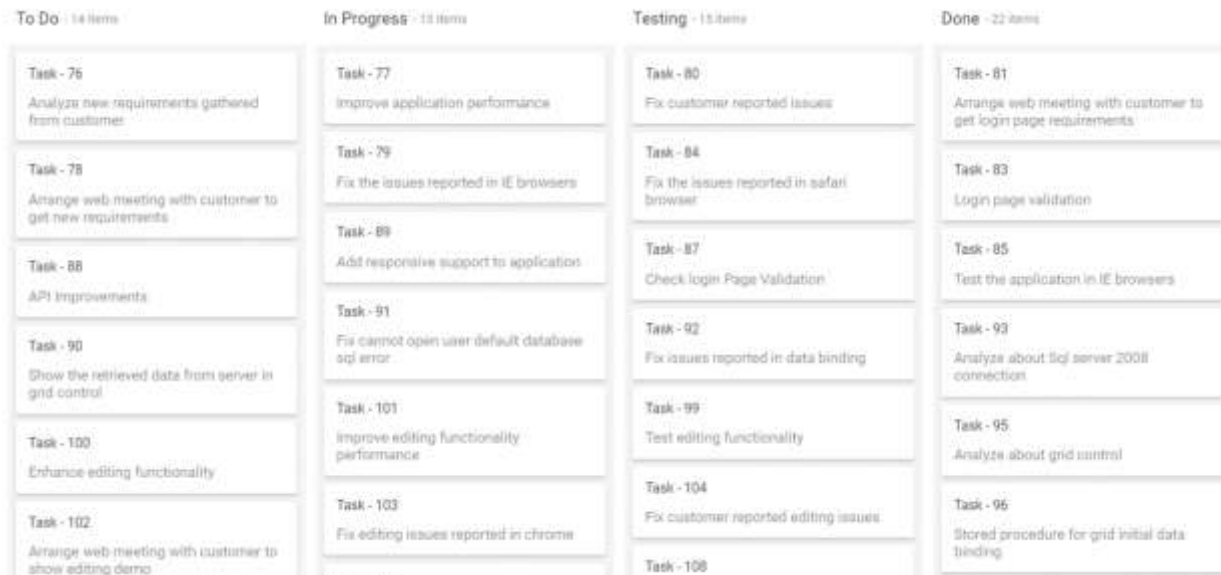
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}

```

Output be like the below.



Sending additional parameters to the server

To add a custom parameter to the data request, use the **addParams** method of **Query** class. Assign the **Query** object with additional parameters to the kanban **Query** property.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).DataSource(dataManger =>
    {
        dataManger.Url("https://ej2services.syncfusion.com/production/web-services/api/Kanban").CrossDomain(true);
    }).Query("new ej.data.Query().addParams('ej2kanban', 'true')").Columns(col=> {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id");
    }).DialogOpen("onDialogOpen").Render()

<script>
    function onDialogOpen(args) {
        args.cancel = true;
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

Note: The parameters added using the **Query** property will be sent along with the data request for every kanban action.

Handling HTTP error

During server interaction from the kanban, some server-side exceptions may occur, and you can acquire those error messages or exception details

in client-side using the **ActionFailure** event.

The argument passed to the **ActionFailure** event contains the error details returned from the server.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").AllowDragAndDrop(false).DataSource(dataManger =>
{
    dataManger.Url("'http://some.com/invalidUrl").CrossDomain(true);
}).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).ActionFailure("onActionFailure").Render()

<script>
function onActionFailure() {
    var span = document.createElement('span');
    this.element.parentNode.insertBefore(span, this.element);
    span.style.color = '#FF0000'
    span.innerHTML = 'Server exception: 404 Not found'
}
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.

Server exception: 404 Not found

Note: The **ActionFailure** event will be triggered not only for the server errors, but also when there is an exception while processing the kanban actions.

Loading data via ajax

You can use Kanban **DataSource** property to bind the datasource to Kanban from external ajax request. In the following code, we have fetched the datasource from the server using ajax request and provided that to the **DataSource** property by using the **OnSuccess** event of ajax.

CSHTML

```
@Html.EJS().Button("btn").Content("ajax").Render()

@Html.EJS().Kanban("kanban").KeyField("ShipCountry").Columns(col=> {
    col.HeaderText("Denmark").KeyField("Denmark").Add();
    col.HeaderText("Brazil").KeyField("Brazil").Add();
    col.HeaderText("Switzerland").KeyField("Switzerland").Add();
    col.HeaderText("Germany").KeyField("Germany").Add();
}).CardSettings(card => {
    card.ContentField("ShippedDate").HeaderField("OrderID");
}).Created("onCreate").Render()

<script>
    function onCreate() {
        var kanbanObj = this;
        var button = document.getElementById('btn');
        button.addEventListener("click", function (e) {
            let ajax = new
ej.base.Ajax("https://ej2services.syncfusion.com/production/web-
services/api/Orders", "GET");
            ajax.send();
            ajax.onSuccess = function (data) {
                kanbanObj.dataSource = JSON.parse(data);
            };
        });
    }
</script>
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.

| AJAX | Denmark - 9 Items | Brazil - 9 Items | Switzerland - 9 Items | Germany - 9 Items |
|------|------------------------------|------------------------------|------------------------------|------------------------------|
| | 10001 1996-07-16T00:00:00 | 10002 1996-09-11T00:00:00 | 10005 1997-12-03T00:00:00 | 10003 1996-10-07T00:00:00 |
| | 10006 1996-07-16T00:00:00 | 10007 1996-09-11T00:00:00 | 10010 1997-12-03T00:00:00 | 10008 1996-10-07T00:00:00 |
| | 10011 1996-07-16T00:00:00 | 10012 1996-09-11T00:00:00 | 10015 1997-12-03T00:00:00 | 10013 1996-10-07T00:00:00 |
| | 10016 1996-07-16T00:00:00 | 10017 1996-09-11T00:00:00 | 10020 1997-12-03T00:00:00 | 10018 1996-10-07T00:00:00 |
| | 10021 1996-07-16T00:00:00 | 10022 1996-09-11T00:00:00 | 10025 1997-12-03T00:00:00 | 10023 1996-10-07T00:00:00 |
| | 10026 1996-07-16T00:00:00 | 10027 1996-09-11T00:00:00 | 10030 1997-12-03T00:00:00 | 10028 1996-10-07T00:00:00 |
| | 10031 1996-07-16T00:00:00 | 10032 1996-09-11T00:00:00 | 10035 1997-12-03T00:00:00 | 10033 1996-10-07T00:00:00 |

Note: * If you bind the DataSource from this way, then it acts like a local dataSource. So you cannot perform any server-side crud actions.

Swimlane in ASP.NET MVC Kanban control

Swimlanes are horizontal categorizations of cards on the Kanban board. It is used for grouping of cards, which brings transparency to the workflow process.

Render swimlane row

Cards can be grouped based on **KeyField** and displayed in rows, which are separated by columns. It is mandatory to define the **KeyField** that is mapped from the datasource for rendering swimlane rows in the Kanban board.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
```



```

public string Title { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the

```

```

customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

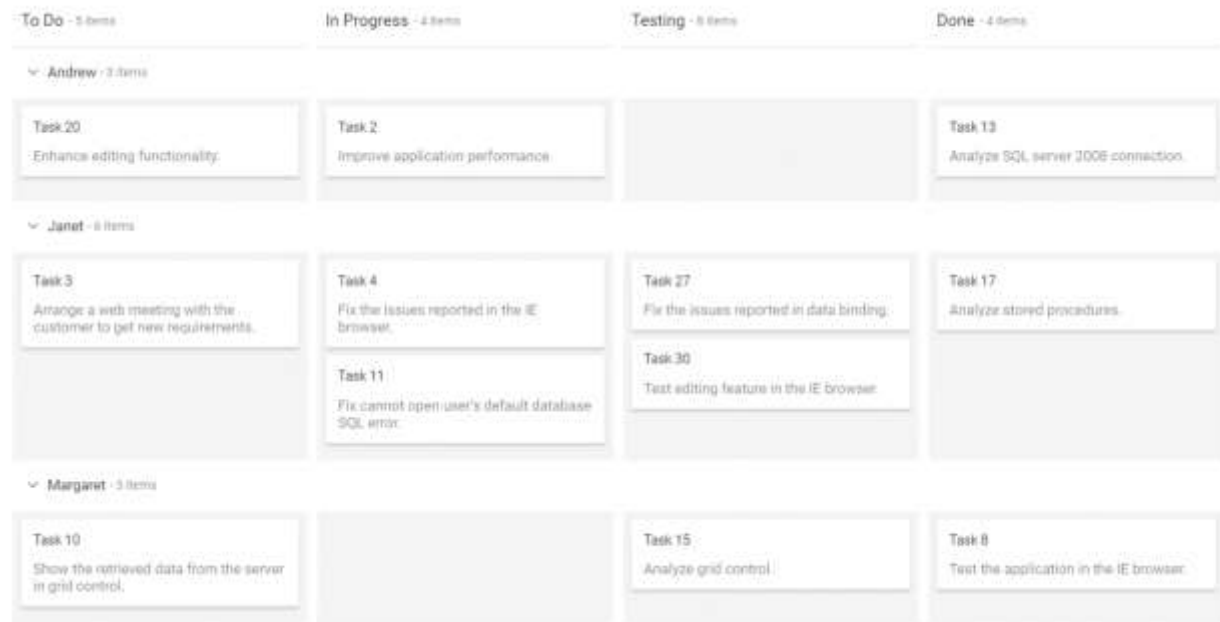
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Custom row text

Customize the swimlane row header text by using the `TextField` property mapped from datasource.

Note: It is not mandatory to define the `TextField` to `SwimlaneSettings`. It will automatically consider the `KeyField` to swimlane row header text.

If the mapping `TextField` key is not present in the datasource, it will consider the swimlane `KeyField` as swimlane row header text.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee").TextField("AssigneeName");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
```

```

public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public string AssigneeName { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", AssigneeName
= "Nancy", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", AssigneeName =
"Andrew", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", AssigneeName
= "Janet", RankId = 2, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", AssigneeName = "Janet",
RankId = 2, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", AssigneeName = "Steven", RankId = 1,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", AssigneeName = "Robert", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", AssigneeName =
"Nancy", RankId = 2, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", AssigneeName = "Margaret", RankId = 3,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation, Fix", Estimate = 1, Assignee = "Steven walker", AssigneeName =
"Steven", RankId = 1, Color = "#8b447a" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", AssigneeName =
"Margaret", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling",
AssigneeName = "Janet", RankId = 4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId
= 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", AssigneeName =
"Andrew", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", AssigneeName = "Margaret", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", AssigneeName = "Margaret", RankId = 5, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker",
AssigneeName = "Steven", RankId = 6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId = 7,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", AssigneeName = "Nancy", RankId = 1, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", AssigneeName = "Nancy", RankId = 5, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", AssigneeName = "Andrew", RankId
= 5, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance

```

```

of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", AssigneeName =
"Nancy", RankId = 5, Color = "#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", AssigneeName
= "Steven", RankId = 6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", AssigneeName =
"Janet", RankId = 6, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", AssigneeName = "Steven", RankId
= 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", AssigneeName =
"Nancy", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", AssigneeName =
"Michael", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId
= 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", AssigneeName = "Nancy", RankId =
5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", AssigneeName = "Robert", RankId =
7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId
= 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{

```

```
public ActionResult Index()
{
    ViewBag.data = new KanbanDataModels().KanbanTasks();
    return View();
}
```

Template

You can customize the Kanban swimlane row by using the **Template** property, which is specified within the **SwimlaneSettings** property. In this demo, the swimlane header is customized with HTML element.

CSHTML

```
<div>

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obje
ct>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {

swim.KeyField("Assignee").Template("#swimlaneTemplate").TextField("AssigneeName");
    }).Render()

</div>
<script id="swimlaneTemplate" type="text/x-jsrender">
    <div class='swimlane-template e-swimlane-template-table'>
        <div class="e-swimlane-row-text"><span>${textField}</span></div>
    </div>
</script>
<style>
    .swimlane-template {
        font-size: 15px;
        font-weight: 500;
    }
    .swimlane-template img {
        height: 24px;
        width: 24px;
        border-radius: 50%;
    }
    .swimlane-template span {
        padding-left: 10px;
    }
</style>
```

DATASOURCE.CS

```
public class KanbanDataModels
```



```

{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public string AssigneeName { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", AssigneeName
= "Nancy", RankId = 1, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", AssigneeName =
"Andrew", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", AssigneeName
= "Janet", RankId = 2, Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", AssigneeName = "Janet",
RankId = 2, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", AssigneeName = "Steven", RankId = 1,
Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", AssigneeName = "Robert", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", AssigneeName =
"Nancy", RankId = 2, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", AssigneeName = "Margaret", RankId = 3,
Color = "#8b447a" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", AssigneeName =
"Steven", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", AssigneeName =
"Margaret", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling",
AssigneeName = "Janet", RankId = 4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId
= 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", AssigneeName =
"Andrew", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", AssigneeName = "Margaret", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", AssigneeName = "Margaret", RankId = 5, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker",
AssigneeName = "Steven", RankId = 6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId = 7,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", AssigneeName = "Nancy", RankId = 1, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", AssigneeName = "Nancy", RankId = 5, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing

```

```

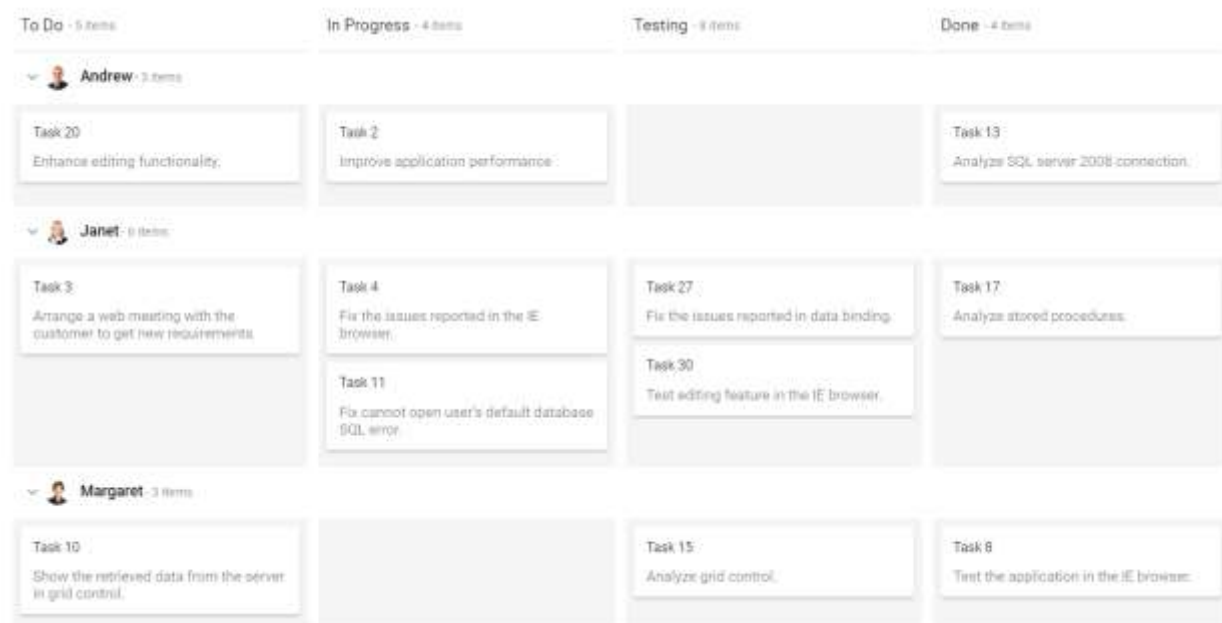
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", AssigneeName = "Andrew", RankId
= 5, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", AssigneeName =
"Nancy", RankId = 5, Color = "#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", AssigneeName
= "Steven", RankId = 6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", AssigneeName =
"Janet", RankId = 6, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", AssigneeName = "Steven", RankId
= 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", AssigneeName =
"Nancy", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", AssigneeName =
"Michael", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId
= 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", AssigneeName = "Nancy", RankId =
5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", AssigneeName = "Robert", RankId =
7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", AssigneeName = "Janet", RankId
= 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.

**Sorting**

Swimlane rows are rendered on descending order when using the **SortBy** property set to **Descending** order. By default, swimlane rows are rendered by **Ascending** order.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee").SortBy(SortType.Descending);
}).Render()
```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE

```

```

browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

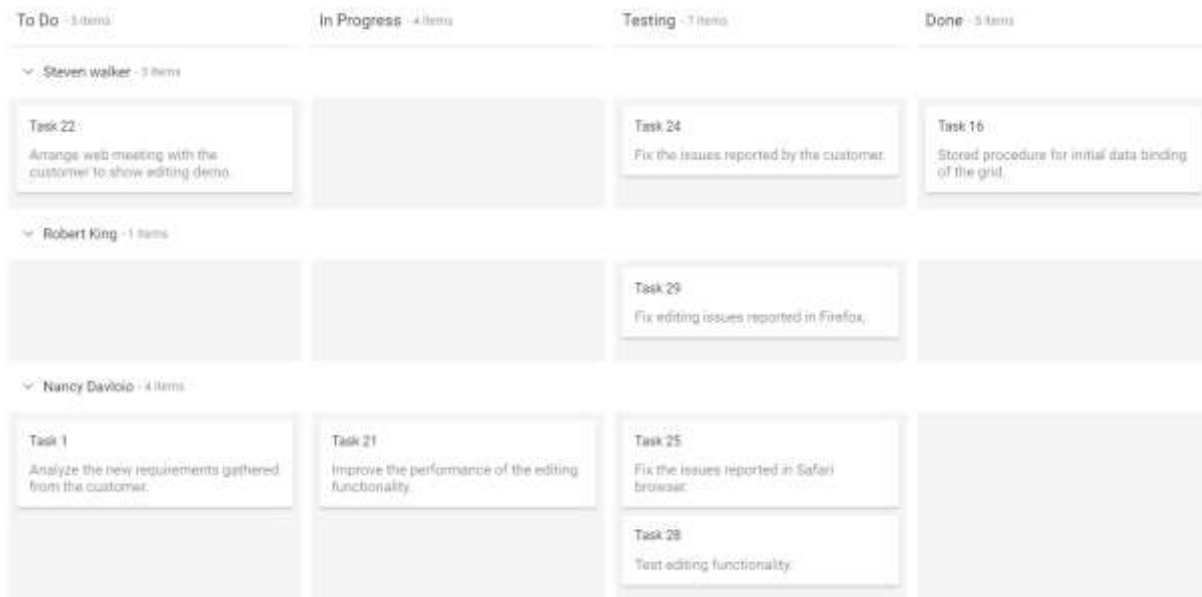
```



```
}

```

Output be like the below.



Drag-and-drop

By default, The Kanban does not allow dragging the cards across the swimlane rows. Enabling the `DragAndDrop` property allows you to drag the cards across the swimlane rows, which is specified inside `SwimlaneSettings` property.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee").AllowDragAndDrop(true);
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
}
```



```

public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
}

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
        "Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
        default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
        "Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
        4, Color = "#cc0000" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
        "Task - 29017", Status = "Review", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
        "Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
        connection.", Type = "Story", Priority = "Release Breaker", Tags =
        "Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
        "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
        "Task - 29019", Status = "Validate", Summary = "Validate databinding
        issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
        1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
        "Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
        "Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
        "Margaret hamilt", RankId = 5, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
        "Task - 29021", Status = "Close", Summary = "Stored procedure for initial
        data binding of the grid.", Type = "Others", Priority = "Release Breaker",
        Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
        6, Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
        "Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
        Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
        = 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
        "Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
        Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
        Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
        "Task - 29024", Status = "Review", Summary = "Test editing functionality.",
        Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
        Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues

```

```

reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Create empty row

You can render the empty swimlane row by enabling the **ShowEmptyRow** property. If mapping **KeyField** does not have cards, empty swimlane row will be rendered.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obje
ct>)ViewBag.data).Columns(col =>

```

```

        {
            col.HeaderText("To Do").KeyField("Open").Add();
            col.HeaderText("In Progress").KeyField("InProgress").Add();
            col.HeaderText("Testing").KeyField("Testing").Add();
            col.HeaderText("Done").KeyField("Close").Add();
        }).CardSettings(card => {
            card.ContentField("Summary").HeaderField("Id");
        }).SwimlaneSettings(swim => {
            swim.KeyField("Assignee").ShowEmptyRow(true);
        }).Render()
    }
}

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",

```

```

Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

```

```

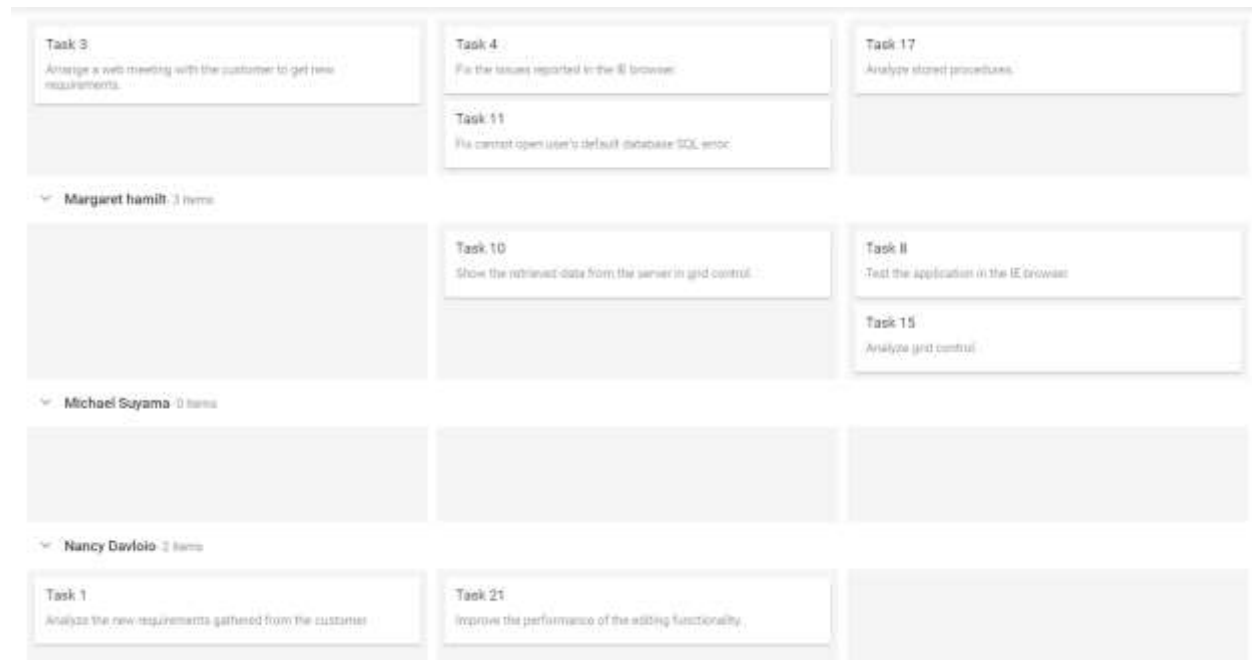
        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
        "Task - 29024", Status = "Review", Summary = "Test editing functionality.",
        Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
        Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.

**Calculate cards count**

Users can show or hide the cards count by swimlane row in header when enabling the **ShowItemCount** property, which is enabled by default on the Kanban board.

Note: Provided localization support for **Items** text.

In below demo, disabled on **ShowItemCount** property on rendering swimlane row without total count.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
```



```
swim.KeyField("Assignee").ShowItemCount(false);
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    }
}
```



```

TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",

```

```

Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });

return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
    }
}

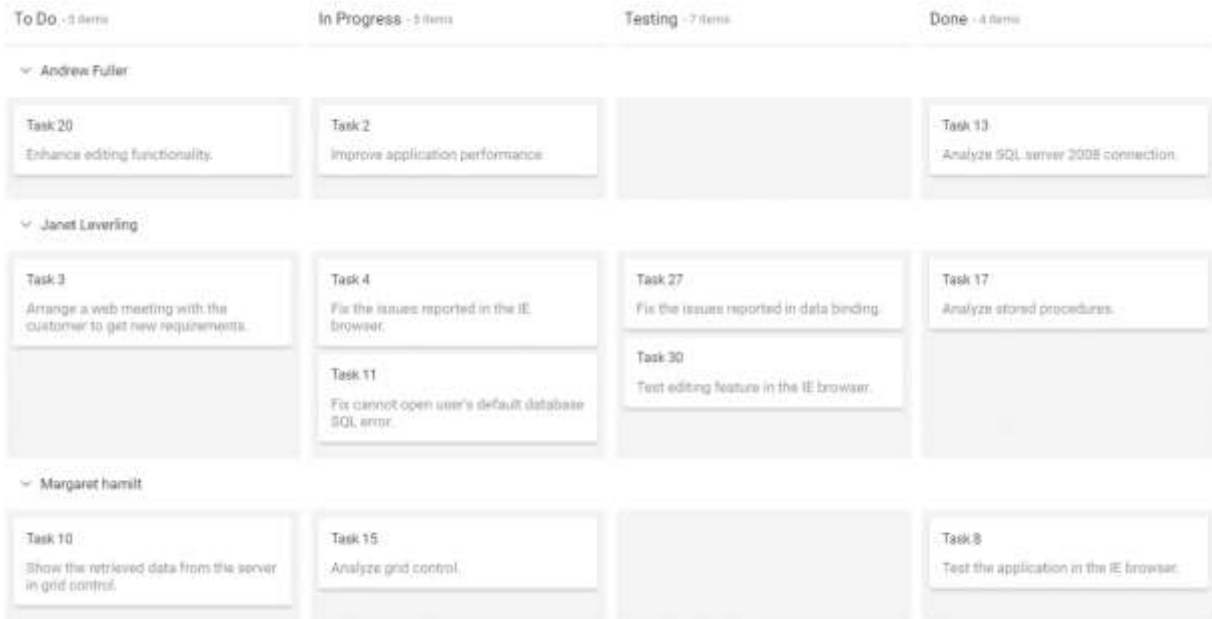
```

```

        return View();
    }
}

```

Output be like the below.



Enable frozen rows

Frozen rows provide an option to make the current swimlane row header text always visible on top of the content while scrolling the Kanban content. The swimlane header text will be changed dynamically, when you scroll to another swimlane row.

By default, the `EnableFrozenRows` property is set as `false`. If you wish to show the swimlane frozen rows, you can enable the `EnableFrozenRows` property.

Note: This feature support only when using Kanban content scrolling.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").Height("500").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}) .CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}) .SwimlaneSettings(swim => {
    swim.KeyField("Assignee").EnableFrozenRows(true);
}) .Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =

```

```

"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

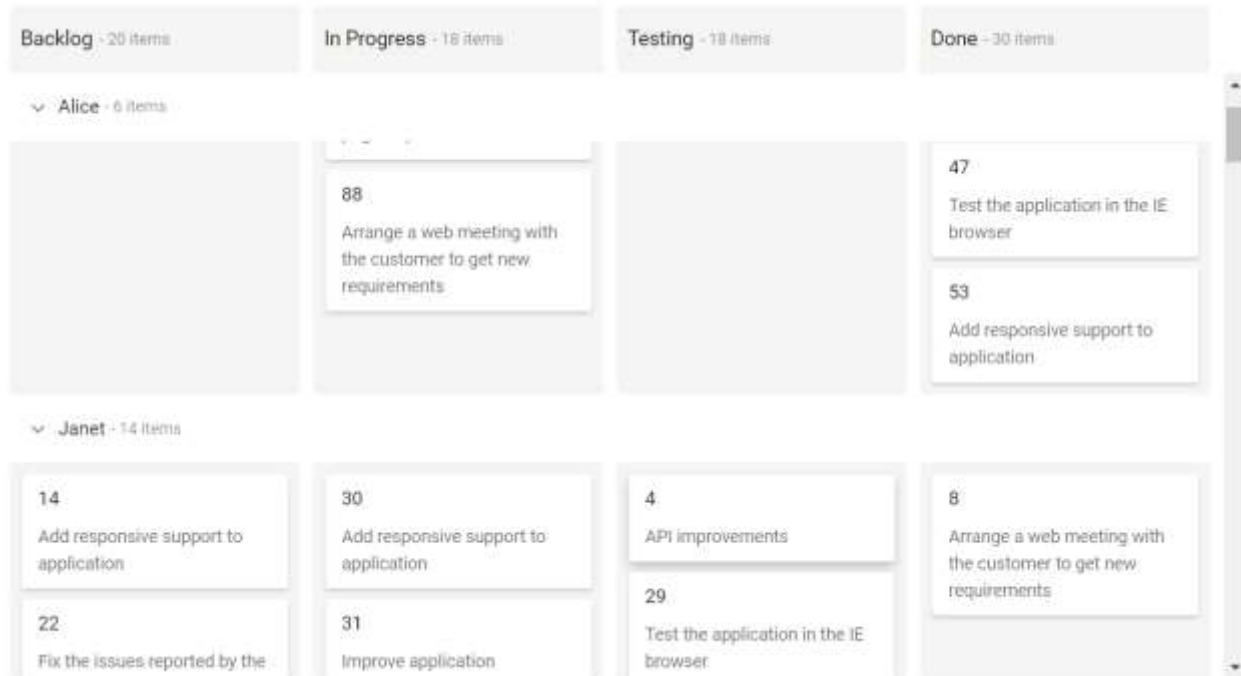
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Drag and drop in ASP.NET MVC Kanban control

All cards can be dragged and dropped across the columns or within the columns or swimlane row or kanban to an external source and vice versa.

The following drag and drop types are available in the Kanban board.

- Internal drag and drop
- Column drag and drop
- Swimlane drag and drop
- External drag and drop
- Kanban to Kanban
- Kanban to External source and vice versa.

Note: Dropped card position varies based on the `SortSettings` property.

Internal drag and drop

Allows the user to drag and drop the cards within the kanban board. Based on this, we can categorize into two ways.

- Column drag and drop
- Swimlane drag and drop

Column drag and drop

By default, all cards can be dragged and dropped across the columns and within the columns. You cannot drag and drop the cards when disabling the `AllowDragAndDrop` property.

Note: You can prevent the drag or drop behavior of the particular column by disabling the `AllowDrag` or `AllowDrop` property.

 You can also control the flow of transition cards between the columns by using the **TransitionColumns** property.

In the following example, disable the drag and drop behavior on the Kanban board.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).AllowDragAndDrop(false).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
```



```

"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Swimlane drag and drop

By default, Swimlane allows drag and drop across the columns within the swimlane row. Kanban does not allow dragging the cards across the swimlane rows.

Enabling the `DragAndDrop` property allows you to drag the cards across the swimlane rows, which is specified inside the `SwimlaneSettings` property.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee").AllowDragAndDrop(true);
}).Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
}

```

```

public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =

```

```

"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

External drag and drop

Allows the user to drag and drop the cards from one kanban to another kanban or Kanban to an external source and vice versa.

Kanban to kanban

Drag and drop the card from one kanban to another kanban and vice versa. This can be achieved by specifying the `ExternalDropId` property which is used to specify the id of the dropped kanban element and the `DragStop` event which is used to delete the card on dragged Kanban and add the card on dropped Kanban using the `deleteCard` and `addCard` public methods.

Note: Before adding a card to dropped kanban, you can manually change the card data **HeaderField** when the same card data **HeaderField** is dropped to another Kanban.

In the following example, Drag the card from one Kanban and drop it into another kanban using the **DragStop** event. In this event, remove the card from the dragged Kanban by using the **deleteCard** public method and add the card to the dropped Kanban by using the **addCard** public method.

CSHTML

```
<div class="container-fluid">
  <div class="row">
    <div class="col-md-6">
      <h4>Kanban A</h4>

@Html.EJS().Kanban("kanbanA").KeyField("Status").DataSource((IEnumerable<obj
ect>)ViewBag.data).ExternalDropId((string[])ViewBag.externalDropIdA).Columns
(col =>
    {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card =>
    {
        card.ContentField("Summary").HeaderField("Id");
    }).DragStop("onDragStopA").Render()

    </div>
    <div class="col-md-6">
      <h4>Kanban B</h4>

@Html.EJS().Kanban("kanbanB").KeyField("Status").DataSource((IEnumerable<obj
ect>)ViewBag.data).ExternalDropId((string[])ViewBag.externalDropIdB).Columns
(col =>
    {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card =>
    {
        card.ContentField("Summary").HeaderField("Id");
    }).DragStop("onDragStopB").Render()

    </div>
  </div>
</div>
<script>
  function onDragStopA(args) {
    var kanbanObjA =
document.querySelector("#kanbanA").ej2_instances[0];
    var kanbanObjB =
document.querySelector("#kanbanB").ej2_instances[0];
    if (ej.base.closest(args.event.target, '#kanbanB')) {
      kanbanObjA.deleteCard(args.data);
      args.data.forEach(function (card, i) {
        var index = kanbanObjB.kanbanData.findIndex(function
(colData) {
          return colData[kanbanObjB.cardSettings.headerField] ===
card[kanbanObjB.cardSettings.headerField];
        });
        if (index !== -1) {
```

```

        card[kanbanObjB.cardSettings.headerField] =
Math.max.apply(Math, kanbanObjB.kanbanData.map(function (obj) { return
parseInt(obj[kanbanObjB.cardSettings.headerField], 10); }))) + ++i;
    }
    });
    kanbanObjB.addCard(args.data, args.dropIndex);
    args.cancel = true;
}
}
function onDragStopB(args) {
    var kanbanObjA =
document.querySelector("#kanbanA").ej2_instances[0];
    var kanbanObjB =
document.querySelector("#kanbanB").ej2_instances[0];
    if (ej.base.closest(args.event.target, '#kanbanA')) {
        kanbanObjB.deleteCard(args.data);
        args.data.forEach(function (card, i) {
            var index = kanbanObjA.kanbanData.findIndex(function
(colData) {
                return colData[kanbanObjA.cardSettings.headerField] ===
card[kanbanObjA.cardSettings.headerField];
            });
            if (index !== -1) {
                card[kanbanObjA.cardSettings.headerField] =
Math.max.apply(Math, kanbanObjA.kanbanData.map(function (obj) { return
parseInt(obj[kanbanObjA.cardSettings.headerField], 10); }))) + ++i;
            }
        });
        kanbanObjA.addCard(args.data, args.dropIndex);
        args.cancel = true;
    }
}
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = 1, Title = "Task -
29001", Status = "Open", Summary = "Analyze the new requirements gathered
from the customer.", Type = "Story", Priority = "Low", Tags =

```



```

"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 2, Title = "Task -
29002", Status = "InProgress", Summary = "Improve application performance",
Type = "Improvement", Priority = "Normal", Tags = "Improvement", Estimate =
6, Assignee = "Andrew Fuller", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = 3, Title = "Task -
29003", Status = "Open", Summary = "Arrange a web meeting with the customer
to get new requirements.", Type = "Others", Priority = "Critical", Tags =
"Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = 4, Title = "Task -
29004", Status = "InProgress", Summary = "Fix the issues reported in the IE
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE", Estimate
= 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = 5, Title = "Task -
29005", Status = "Review", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate =
3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = 6, Title = "Task -
29007", Status = "Validate", Summary = "Validate new requirements", Type =
"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = 7, Title = "Task -
29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix, Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});
    TaskDetails.Add(new KanbanDataModels { Id = 8, Title = "Task -
29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 9, Title = "Task -
29011", Status = "Validate", Summary = "Validate the issues reported by the
customer.", Type = "Story", Priority = "High", Tags = "Validation, Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 10, Title = "Task -
29015", Status = "Open", Summary = "Show the retrieved data from the server
in grid control.", Type = "Story", Priority = "High", Tags = "Database, SQL",
Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = 11, Title = "Task -
29016", Status = "InProgress", Summary = "Fix cannot open user's default
database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database, Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = 12, Title = "Task -
29017", Status = "Review", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = 13, Title = "Task -
29018", Status = "Close", Summary = "Analyze SQL server 2008 connection.",
Type = "Story", Priority = "Release Breaker", Tags = "Grid, Sql", Estimate =
2, Assignee = "Andrew Fuller", RankId = 4, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 14, Title = "Task -
29019", Status = "Validate", Summary = "Validate databinding issues.", Type

```

```

= "Story", Priority = "Low", Tags = "Validation", Estimate = 1.5, Assignee =
"Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 15, Title = "Task -
29020", Status = "Close", Summary = "Analyze grid control.", Type = "Story",
Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee = "Margaret
hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 16, Title = "Task -
29021", Status = "Close", Summary = "Stored procedure for initial data
binding of the grid.", Type = "Others", Priority = "Release Breaker", Tags =
"Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId = 6, Color
= "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = 17, Title = "Task -
29022", Status = "Close", Summary = "Analyze stored procedures.", Type =
"Story", Priority = "Release Breaker", Tags = "Procedures", Estimate = 5.5,
Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 18, Title = "Task -
29023", Status = "Validate", Summary = "Validate editing issues.", Type =
"Story", Priority = "Critical", Tags = "Editing", Estimate = 1, Assignee =
"Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 19, Title = "Task -
29024", Status = "Review", Summary = "Test editing functionality.", Type =
"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = 20, Title = "Task -
29025", Status = "Open", Summary = "Enhance editing functionality.", Type =
"Improvement", Priority = "Low", Tags = "Editing", Estimate = 3.5, Assignee
= "Andrew Fuller", RankId = 5, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = 21, Title = "Task -
29026", Status = "InProgress", Summary = "Improve the performance of the
editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = 22, Title = "Task -
29027", Status = "Open", Summary = "Arrange web meeting with the customer to
show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = 23, Title = "Task -
29029", Status = "Review", Summary = "Fix the editing issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Editing,Fix", Estimate =
3.5, Assignee = "Janet Leverling", RankId = 6, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = 24, Title = "Task -
29030", Status = "Testing", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Critical", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = 25, Title = "Task -
29031", Status = "Testing", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = 26, Title = "Task -
29032", Status = "Testing", Summary = "Check Login page validation.", Type =
"Story", Priority = "Release Breaker", Tags = "Testing", Estimate = 0.5,
Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = 27, Title = "Task -
29033", Status = "Testing", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = 28, Title = "Task - 29034", Status = "Testing", Summary = "Test editing functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = 29, Title = "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix", Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = 30, Title = "Task - 29036", Status = "Testing", Summary = "Test editing feature in the IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.externalDropIdA = new string[] { "#kanbanB" };
        ViewBag.externalDropIdB = new string[] { "#kanbanA" };
        ViewBag.data = new KanbanDataModels().KanbanTasks();
    }
}

```

Treeview to Kanban

Drag the card from the Kanban board and drop it to the Treeview component and vice versa.

In the following sample, remove the data from the Kanban board using the `deleteCard` public method and add to the Treeview component using the `addNodes` public method at Kanban `DragStop` event when dragging the card and dropping it to the Treeview component. Remove the data from Treeview using the `removeNodes` public method and add to Kanban board using the `openDialog` public method when dragging the list from the Treeview component and dropping it to the kanban board.

CSHTML

```

<div class="container-fluid">
    <div class="row">
        <div class="col-md-6">
            <h4>Kanban</h4>

            @Html.EJS().Kanban("Kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).ExternalDropId((string[])ViewBag.externalDropId).Columns(col =>
            {
                col.HeaderText("To Do").KeyField("Open").Add();
                col.HeaderText("Done").KeyField("Close").Add();
            }).CardSettings(card =>
            {
                card.ContentField("Summary").HeaderField("Id");
            }).DragStop("onDragStop").Render()

        </div>
    </div>

```

```

        <div class="col-md-6">
            <h4>TreeView</h4>
            @Html.EJS().TreeView("TreeView").Fields(fields =>
fields.DataSource((IEnumerable<object>)ViewBag.treeDataSource).Id("Id").Text
("Status")).AllowDragAndDrop(true).NodeDragStop("onTreeDragStop").NodeTempla
te("#treeTemplate").Render()
            </div>
        </div>
</div>
<script>
function onDragStop(args) {
    var kanbanObj = document.querySelector("#Kanban").ej2_instances[0];
    var treeObj = document.querySelector("#TreeView").ej2_instances[0];
    if (ej.base.closest(args.event.target, "#TreeView")) {
        kanbanObj.deleteCard(args.data);
        treeObj.addNodes(args.data);
        args.cancel = true;
    }
}
function onTreeDragStop(args) {
    var kanbanObj = document.querySelector("#Kanban").ej2_instances[0];
    var treeObj = document.querySelector("#TreeView").ej2_instances[0];
    if (ej.base.closest(args.event.target, '#Kanban')) {
        var treeData = treeObj.fields.dataSource;
        var filteredData = treeData.filter(function (item) { return
item.Id === parseInt(args.draggedNodeData.id, 10); });
        treeObj.removeNodes([args.draggedNodeData.id]);
        kanbanObj.openDialog('Add', filteredData[0]);
        args.cancel = true;
    }
}
</script>
<script id="treeTemplate" type="text/x-template">
    <div id="waiting">
        <div id="treelist">
            <div id="treeviewlist">${Id} - ${Status}</div>
        </div>
    </div>
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> TreeDataSource()

```

```

    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = 6, Title = "Task -
29007", Status = "Validate", Summary = "Validate new requirements", Type =
"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = 7, Title = "Task -
29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});
        TaskDetails.Add(new KanbanDataModels { Id = 8, Title = "Task -
29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = 9, Title = "Task -
29011", Status = "Validate", Summary = "Validate the issues reported by the
customer.", Type = "Story", Priority = "High", Tags = "Validation,Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = 10, Title = "Task -
29015", Status = "Open", Summary = "Show the retrieved data from the server
in grid control.", Type = "Story", Priority = "High", Tags = "Database,SQL",
Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4, Color = "#8b447a"
});
        return TaskDetails;
    }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = 1, Title = "Task -
29001", Status = "Open", Summary = "Analyze the new requirements gathered
from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = 2, Title = "Task -
29002", Status = "InProgress", Summary = "Improve application performance",
Type = "Improvement", Priority = "Normal", Tags = "Improvement", Estimate =
6, Assignee = "Andrew Fuller", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = 3, Title = "Task -
29003", Status = "Open", Summary = "Arrange a web meeting with the customer
to get new requirements.", Type = "Others", Priority = "Critical", Tags =
"Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = 4, Title = "Task -
29004", Status = "InProgress", Summary = "Fix the issues reported in the IE
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE", Estimate
= 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = 5, Title = "Task -
29005", Status = "Review", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate =
3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.externalDropId = new string[] { "#TreeView" };
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        ViewBag.treeDataSource = new KanbanDataModels().TreeDataSource();
        return View();
    }
}
```

Schedule to Kanban

Drag the card from the Kanban board and drop it to the Schedule component and vice versa.

In the following sample, remove the data from the Kanban board using the `deleteCard` public method and add to the schedule component using the `addNodes` public method at Kanban `DragStop` event when dragging the card and dropping it to the Treeview component. Remove the data from Treeview using the `removeNodes` public method and add to Kanban board using the `addCard` public method when dragging the list from the Treeview component and dropping it to the kanban board.

CSHTML

```
@using Syncfusion.EJ2.Schedule
<div class="container-fluid">
    <div class="row">
        <div class="col-md-6">
            <h4>Kanban</h4>

            @Html.EJS().Kanban("Kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).ExternalDropId((string[])ViewBag.externalDropId).Columns(col =>
            {
                col.HeaderText("To Do").KeyField("Open").Add();
                col.HeaderText("Done").KeyField("Close").Add();
            }).CardSettings(card =>
            {
                card.ContentField("Summary").HeaderField("Id");
            }).DragStop("onDragStop").Render()

        </div>
        <div class="col-md-6">
            <h4>Schedule</h4>

            @Html.EJS().Schedule("Schedule").Width("100%").Height("650px").Views(view =>
            {
                view.Option(View.TimelineDay).Add();
                view.Option(View.TimelineMonth).Add();
            }).CurrentView(View.TimelineDay).SelectedDate(new DateTime(2018, 7, 1)).WorkHours(new ScheduleWorkHours { Highlight = true, Start = "08:00", End = "18:00" }).Group(group =>
            group.EnableCompactView(false).Resources(ViewBag.Resources)).Resources(res =>
            {
```

```

res.DataSource(ViewBag.Departments).Field("DepartmentID").Title("Department")
).Name("Departments").TextField("text").IdField("id").ColorField("color").Add();

res.DataSource(ViewBag.Consultants).Field("ConsultantID").Title("Consultant")
).Name("Consultants").TextField("text").IdField("id").ColorField("color").GroupIDField("groupId").AllowMultiple(false).Add();
}).ResourceHeaderTemplate("#resource-template").DragStop("scheduleDragStop").EventSettings(e => e.Fields(f =>
f.Subject(sub => sub.Name("Name").Title("Patient Name")).StartTime(start =>
start.Name("StartTime").Title("From")).EndTime(end =>
end.Name("EndTime").Title("To")).Description(des =>
des.Name("Description").Title("Reason"))).DataSource(ViewBag.datasources)).Render()

</div>
</div>
</div>
<!-- Resource Header Template -->
<script id="resource-template" type="text/x-template">
    <div class="template-wrap">
        <div class="specialist-category">
            ${getConsultantImage(data)}
            <div class="specialist-name">${getConsultantName(data)}</div>
            <div class="specialist-
designation">${getConsultantDesignation(data)}</div>
        </div>
    </div>
</script>
<script>
    function getConsultantName(value) {
        return value.resourceData[value.resource.textField];
    };
    function getConsultantImage() {
        return '';
    };
    function getConsultantDesignation(value) {
        let resourceName = value.resourceData[value.resource.textField];
        if (resourceName === 'GENERAL' || resourceName === 'DENTAL') {
            return '';
        } else {
            return value.resourceData.designation;
        }
    };
    function scheduleDragStop(args) {
        var kanbanObj = document.querySelector("#Kanban").ej2_instances[0];
        var scheduleObj =
document.querySelector("#Schedule").ej2_instances[0];
        if (ej.base.closest(args.event.target, '#Kanban')) {
            scheduleObj.deleteEvent(args.data.Id);
            ej.base.removeClass([scheduleObj.element.querySelector('.e-
selected-cell')], 'e-selected-cell');
            kanbanObj.openDialog('Add', args.data);
            args.cancel = true;
        }
    }
    function onDragStop(args) {

```

```

        var kanbanObj = document.querySelector("#Kanban").ej2_instances[0];
        var scheduleObj =
document.querySelector("#Schedule").ej2_instances[0];
        if (ej.base.closest(args.event.target, '#Schedule')) {
            kanbanObj.deleteCard(args.data);
            scheduleObj.openEditor(args.data[0], 'Add', true);
            args.cancel = true;
        }
    }
}
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = 1, Title = "Task -
29001", Status = "Open", Summary = "Analyze the new requirements gathered
from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = 2, Title = "Task -
29002", Status = "InProgress", Summary = "Improve application performance",
Type = "Improvement", Priority = "Normal", Tags = "Improvement", Estimate =
6, Assignee = "Andrew Fuller", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = 3, Title = "Task -
29003", Status = "Open", Summary = "Arrange a web meeting with the customer
to get new requirements.", Type = "Others", Priority = "Critical", Tags =
"Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = 4, Title = "Task -
29004", Status = "InProgress", Summary = "Fix the issues reported in the IE
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE", Estimate
= 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = 5, Title = "Task -
29005", Status = "Review", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate =
3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        return TaskDetails;
    }
}

public class ScheduleData

```



```

{
    public List<HospitalData> GetHospitalData()
    {
        List<HospitalData> hospitalData = new List<HospitalData>();
        hospitalData.Add(new HospitalData
        {
            Id = 10,
            Name = "David",
            StartTime = new DateTime(2018, 7, 1, 9, 0, 0),
            EndTime = new DateTime(2018, 7, 1, 10, 0, 0),
            Description = "Health Checkup",
            DepartmentID = 1,
            ConsultantID = 1,
            DepartmentName = "GENERAL"
        });
        hospitalData.Add(new HospitalData
        {
            Id = 11,
            Name = "John",
            StartTime = new DateTime(2018, 7, 1, 10, 30, 0),
            EndTime = new DateTime(2018, 7, 1, 11, 30, 0),
            Description = "Tooth Erosion",
            DepartmentID = 2,
            ConsultantID = 2,
            DepartmentName = "DENTAL"
        });
        hospitalData.Add(new HospitalData
        {
            Id = 12,
            Name = "Peter",
            StartTime = new DateTime(2018, 7, 1, 12, 0, 0),
            EndTime = new DateTime(2018, 7, 1, 13, 0, 0),
            Description = "Eye and Spectacles Checkup",
            DepartmentID = 1,
            ConsultantID = 3,
            DepartmentName = "GENERAL"
        });
        hospitalData.Add(new HospitalData
        {
            Id = 13,
            Name = "Starc",
            StartTime = new DateTime(2018, 7, 1, 14, 0, 0),
            EndTime = new DateTime(2018, 7, 1, 15, 0, 0),
            Description = "Toothaches",
            DepartmentID = 2,
            ConsultantID = 4,
            DepartmentName = "DENTAL"
        });
        hospitalData.Add(new HospitalData
        {
            Id = 14,
            Name = "James",
            StartTime = new DateTime(2018, 7, 1, 10, 0, 0),
            EndTime = new DateTime(2018, 7, 1, 11, 0, 0),
            Description = "Surgery Appointment",
            DepartmentID = 1,
            ConsultantID = 5,

```

```

        DepartmentName = "GENERAL"
    });
hospitalData.Add(new HospitalData
{
    Id = 15,
    Name = "Jersey",
    StartTime = new DateTime(2018, 7, 1, 9, 30, 0),
    EndTime = new DateTime(2018, 7, 1, 10, 30, 0),
    Description = "Tooth Sensitivity",
    DepartmentID = 2,
    ConsultantID = 6,
    DepartmentName = "DENTAL"
});
hospitalData.Add(new HospitalData
{
    Id = 16,
    Name = "Albert",
    StartTime = new DateTime(2018, 7, 2, 10, 0, 0),
    EndTime = new DateTime(2018, 7, 2, 11, 30, 0),
    Description = "Skin care treatment",
    DepartmentID = 1,
    ConsultantID = 7,
    DepartmentName = "GENERAL"
});
hospitalData.Add(new HospitalData
{
    Id = 17,
    Name = "Louis",
    StartTime = new DateTime(2018, 7, 2, 12, 30, 0),
    EndTime = new DateTime(2018, 7, 2, 13, 30, 0),
    Description = "General Checkup",
    DepartmentID = 1,
    ConsultantID = 9,
    DepartmentName = "DENTAL"
});
hospitalData.Add(new HospitalData
{
    Id = 18,
    Name = "Williams",
    StartTime = new DateTime(2018, 7, 2, 12, 0, 0),
    EndTime = new DateTime(2018, 7, 2, 14, 0, 0),
    Description = "Mouth Sores",
    DepartmentID = 2,
    ConsultantID = 10,
    DepartmentName = "DENTAL"
});
hospitalData.Add(new HospitalData
{
    Id = 19,
    Name = "David",
    StartTime = new DateTime(2018, 7, 2, 16, 30, 0),
    EndTime = new DateTime(2018, 7, 2, 18, 45, 0),
    Description = "Eye checkup and Treatment",
    DepartmentID = 1,
    ConsultantID = 1,
    DepartmentName = "GENERAL"
});

```

```

        hospitalData.Add(new HospitalData
        {
            Id = 20,
            Name = "John",
            StartTime = new DateTime(2018, 7, 2, 19, 30, 0),
            EndTime = new DateTime(2018, 7, 2, 21, 45, 0),
            Description = "Tooth Decay",
            DepartmentID = 2,
            ConsultantID = 2,
            DepartmentName = "DENTAL"
        });
        return hospitalData;
    }
}

public class HospitalData
{
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime StartTime { get; set; }
    public DateTime EndTime { get; set; }
    public string Description { get; set; }
    public int DepartmentID { get; set; }
    public int ConsultantID { get; set; }
    public string DepartmentName { get; set; }
}

public class ResourceDataSourceModel
{
    public string text { set; get; }
    public int id { set; get; }
    public string color { set; get; }
    public int groupId { set; get; }
}

public class ConsultantDataSourceModel
{
    public string text { set; get; }
    public int id { set; get; }
    public int groupId { set; get; }
    public string color { set; get; }
    public string designation { set; get; }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.externalDropId = new string[] { "#Schedule" };
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        ViewBag.datasources = new ScheduleData().GetHospitalData();
        List<ResourceDataSourceModel> departments = new
        List<ResourceDataSourceModel>();
        departments.Add(new ResourceDataSourceModel { text = "GENERAL", id =
        1, color = "#bbdc00" });
        departments.Add(new ResourceDataSourceModel { text = "DENTAL", id =
        2, color = "#9e5fff" });
    }
}

```

```

        ViewBag.Departments = departments;
        List<ConsultantDataSourceModel> consultants = new
List<ConsultantDataSourceModel>();
        consultants.Add(new ConsultantDataSourceModel { text = "Alice", id =
1, groupId = 1, color = "#bbdc00", designation = "Cardiologist" });
        consultants.Add(new ConsultantDataSourceModel { text = "Nancy", id =
2, groupId = 2, color = "#9e5fff", designation = "Orthodontist" });
        consultants.Add(new ConsultantDataSourceModel { text = "Robert", id
= 3, groupId = 1, color = "#bbdc00", designation = "Optometrist" });
        consultants.Add(new ConsultantDataSourceModel { text = "Robson", id
= 4, groupId = 2, color = "#9e5fff", designation = "Periodontist" });
        consultants.Add(new ConsultantDataSourceModel { text = "Laura", id =
5, groupId = 1, color = "#bbdc00", designation = "Orthopedic" });
        consultants.Add(new ConsultantDataSourceModel { text = "Margaret",
id = 6, groupId = 2, color = "#9e5fff", designation = "Endodontist" });
        ViewBag.Consultants = consultants;
        ViewBag.Resources = new string[] { "Departments", "Consultants" };
        return View();
    }
}

```

Sort in ASP.NET MVC Kanban control

The Kanban provides built-in support to arrange the cards in their columns based on the JSON data order and drop the cards in the columns based on the dropped clone. Initially, users can change the arrangement of cards in the columns and position of the dropped card by using the [sortBy](#) property. The [sortBy](#) property contains three enumeration values as follows.

- Index
- DataSourceOrder
- Custom

Index

SortBy [Index](#) property can be used with or without [field](#) mapping.

Index without field mapping

By default, SortBy [Index](#) property support without any [field](#) mapping. In this behavior, cards are loaded based on the JSON data order and cards are dropped based on the dropped clone.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obje
ct>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =

```

```

"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Index with field mapping

SortBy `Index` property also supports with `field` mapping. In this behavior, cards are loaded based on mapping `field` values, and cards are dropped based on the dropped clone.

Cards are placed in a particular position in the columns where you can drop the cards by specifying the `field` property, which is mapped from the data source. This property allows the users to drop the cards in the Kanban board where the dropped clone is created exactly. It is also helpful to render the cards based on the `field` property value.

Note: The `field` property mapping key value must be in `number` format.

The following cases will dynamically change their `field` value when dropping the cards.

- If the cell has no cards, the dropped card `field` value does not change.
- If the cell has one card and dropped a card to the last position or previous/next cards that do not have continuous order, then the dropped card `field` value will be changed based on their previous card value.
- If the cell has one card and dropped a card on the previous position, then it will compare both the values, and the dropped card `field` value will be changed if the cards have continuous order otherwise values will not be changed.
- When the previous and next cards do not have continuous order, the dropped card `field` value will be changed based on the previous card value.
- When the previous and next cards have continuous order or odd/even value, then the `field` value of the dropped card and the cards followed by the dropped card will be changed based on the **previous** card value with continuous order.

For Example,

Continuous Order -

Consider, Column A has Card A with priority value `1`, Card B with priority value `2`, and Card C with priority value `3` and Column B has Card D with priority value `5`, then the dropped Card D will be placed between Card A and Card B. Now, the Cards D, B, and C will be dynamically changed to the priority values as `2`, `3`, and `4` respectively.

Odd/Even order -

Consider, Column A has Card A with priority value `1`, Card B with priority value `3`, and Card C with priority value `5` and Column B has Card D with priority value `5`, then the Dropped Card D will be placed between Card A and Card B. Now, the Cards D, B, and C will be dynamically changed to the priority values as `2`, `3`, and `5` respectively.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
```



```

        card.ContentField("Summary").HeaderField("Id");
    }).SortSettings(sort => {
sort.SortBy(Syncfusion.EJ2.Kanban.SortOrderBy.Index).Field("RankId");
}).Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =

```

```

"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });

        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{

```

```

public ActionResult Index()
{
    ViewBag.data = new KanbanDataModels().KanbanTasks();
    return View();
}

```

DataSource Order

The SortBy **DataSourceOrder** property does not require any [field](#) mapping. In this behavior, cards are loaded based on the JSON data order, and also cards are dropped based on the JSON data order.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SortSettings(sort => {
    sort.SortBy(Syncfusion.EJ2.Kanban.SortOrderBy.DataSourceOrder);
}).Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Custom

Custom with field mapping

The `SortBy Custom` property must require datasource [field](#) mapping. In this behavior, cards are loaded based on the [field](#) mapping value and also cards are dropped based on the [field](#) mapping value.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obje
ct>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SortSettings(sort => {
    sort.SortBy(Syncfusion.EJ2.Kanban.SortOrderBy.Custom).Field("Summary");
}).Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
}

```



```

public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from

```



```

the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =

```

```

"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Change the direction

Kanban board also provides support for aligning the cards in the columns using the [direction](#) property inside the [sortSettings](#) property. Based on this, cards can be aligned in the columns either in [Ascending](#) or [Descending](#) order. Sorting direction will be performed based on [sortBy](#) property.

Note: By default, cards are aligned in the columns based on **Ascending** order.

In the following sample, cards are aligned in **Descending** order.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SortSettings(sort => {
    sort.SortBy(Syncfusion.EJ2.Kanban.SortOrderBy.Custom).Field("Summary").Direction(Syncfusion.EJ2.Kanban.SortDirection.Descending);
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
```

```

"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Card Editing in ASP.NET MVC Kanban control

The Kanban provides built-in support to add, edit and delete a card using dialog module. User can edit a card using the following ways.

- Built-in dialog module
- Custom Fields
- Dialog template

Default Dialog

When double-click on the cards, the dialog is opened with below fields to edit a card. This dialog contains **Delete**, **Save** and **Cancel** buttons.

- To edit a card, modify the card details and click the **Save** button.
- To delete a card, click **Delete** button.
- Click on the **Cancel** button to cancel the editing action.

The dialog displays with the following fields which mapped to dialog fields by default.

|Key|Type|Text|

|---|----|----|

cardSettings.headerField | Input | ID

keyField | DropDown | -

cardSettings.contentField | TextArea | -

cardSettings.priority(If applicable) | Numeric | -

swimlaneSettings.keyField(If applicable) | DropDown | -

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title = "Task - 29001", Status = "Open", Summary = "Analyze the new requirements gathered from the customer.", Type = "Story", Priority = "Low", Tags = "Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title = "Task - 29002", Status = "InProgress", Summary = "Improve application performance", Type = "Improvement", Priority = "Normal", Tags = "Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title = "Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the customer to get new requirements.", Type = "Others", Priority = "Critical", Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2, Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title = "Task - 29004", Status = "InProgress", Summary = "Fix the issues reported in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title = "Task - 29005", Status = "Review", Summary = "Fix the issues reported by the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate = 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title = "Task - 29007", Status = "Validate", Summary = "Validate new requirements", Type =
```



```

"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title = "Task
- 29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title = "Task
- 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title = "Task
- 29011", Status = "Validate", Summary = "Validate the issues reported by
the customer.", Type = "Story", Priority = "High", Tags = "Validation,Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title = "Task
- 29015", Status = "Open", Summary = "Show the retrieved data from the
server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title = "Task
- 29016", Status = "InProgress", Summary = "Fix cannot open user's default
database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title = "Task
- 29017", Status = "Review", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title = "Task
- 29018", Status = "Close", Summary = "Analyze SQL server 2008 connection.",
Type = "Story", Priority = "Release Breaker", Tags = "Grid,Sql", Estimate =
2, Assignee = "Andrew Fuller", RankId = 4, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title = "Task
- 29019", Status = "Validate", Summary = "Validate databinding issues.",
Type = "Story", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title = "Task
- 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title = "Task
- 29021", Status = "Close", Summary = "Stored procedure for initial data
binding of the grid.", Type = "Others", Priority = "Release Breaker", Tags =
"Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId = 6, Color
= "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title = "Task
- 29022", Status = "Close", Summary = "Analyze stored procedures.", Type =
"Story", Priority = "Release Breaker", Tags = "Procedures", Estimate = 5.5,
Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title = "Task
- 29023", Status = "Validate", Summary = "Validate editing issues.", Type =
"Story", Priority = "Critical", Tags = "Editing", Estimate = 1, Assignee =
"Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title = "Task
- 29024", Status = "Review", Summary = "Test editing functionality.", Type =

```



```

"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title = "Task
- 29025", Status = "Open", Summary = "Enhance editing functionality.", Type
= "Improvement", Priority = "Low", Tags = "Editing", Estimate = 3.5,
Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title = "Task
- 29026", Status = "InProgress", Summary = "Improve the performance of the
editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title = "Task
- 29027", Status = "Open", Summary = "Arrange web meeting with the customer
to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title = "Task
- 29029", Status = "Review", Summary = "Fix the editing issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Editing,Fix",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6, Color = "#cc0000"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title = "Task
- 29030", Status = "Testing", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Critical", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title = "Task
- 29031", Status = "Testing", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title = "Task
- 29032", Status = "Testing", Summary = "Check Login page validation.", Type
= "Story", Priority = "Release Breaker", Tags = "Testing", Estimate = 0.5,
Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title = "Task
- 29033", Status = "Testing", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title = "Task
- 29034", Status = "Testing", Summary = "Test editing functionality.", Type
= "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title = "Task
- 29035", Status = "Testing", Summary = "Fix editing issues reported in
Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title = "Task
- 29036", Status = "Testing", Summary = "Test editing feature in the IE
browser.", Type = "Story", Priority = "Normal", Tags = "Testing", Estimate =
5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

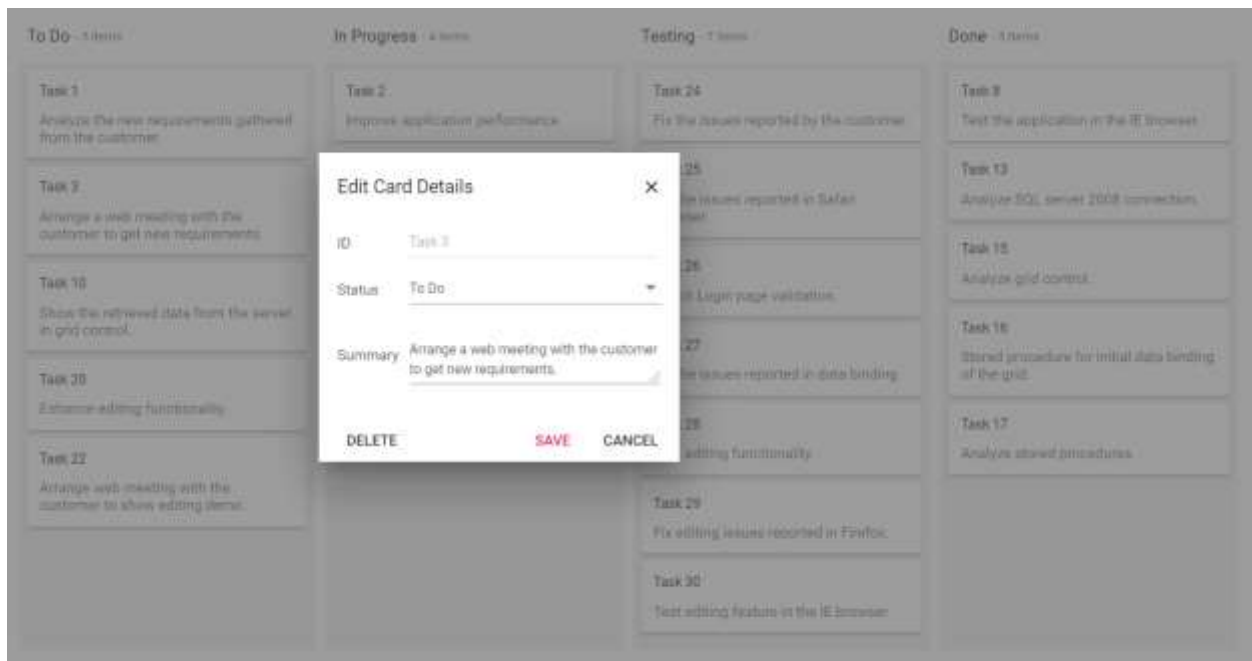
```

CONTROLLER.CS

```
public class HomeController : Controller
```

```
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.



Custom Fields

You can change the default fields of dialog using **Fields** property inside the **DialogSettings** property. The **Key** property used to map the **DataSource** value and rendered the corresponding component based on specified **Type** property.

The following types are available in dialog fields.

- String
- Numeric
- TextArea
- DropDown
- TextBox
- Input

Note: If **Type** is not defined in the fields, then it renders as the HTML input element in dialog.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obj>ct>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
```

```

col.HeaderText("In Progress").KeyField("InProgress").Add();
col.HeaderText("Testing").KeyField("Testing").Add();
col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DialogSettings((item) => item.Fields(ViewBag.dialog)).Render()

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in

```

```

Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });

        return TaskDetails;
    }

}

public class KanbanDialogModels
{
    public string text { get; set; }
    public string key { get; set; }
    public string type { get; set; }
}

```

```

public List<KanbanDialogModels> DialogCards()
{
    List<KanbanDialogModels> DialogCard = new
List<KanbanDialogModels>();
    DialogCard.Add(new KanbanDialogModels { key = "Id", type =
"Input" });
    DialogCard.Add(new KanbanDialogModels { key = "Status", type =
"DropDown" });
    DialogCard.Add(new KanbanDialogModels { key = "Estimate", type =
"Numeric" });
    DialogCard.Add(new KanbanDialogModels { key = "Summary", type =
"TextArea" });
    return DialogCard;
}
}

```

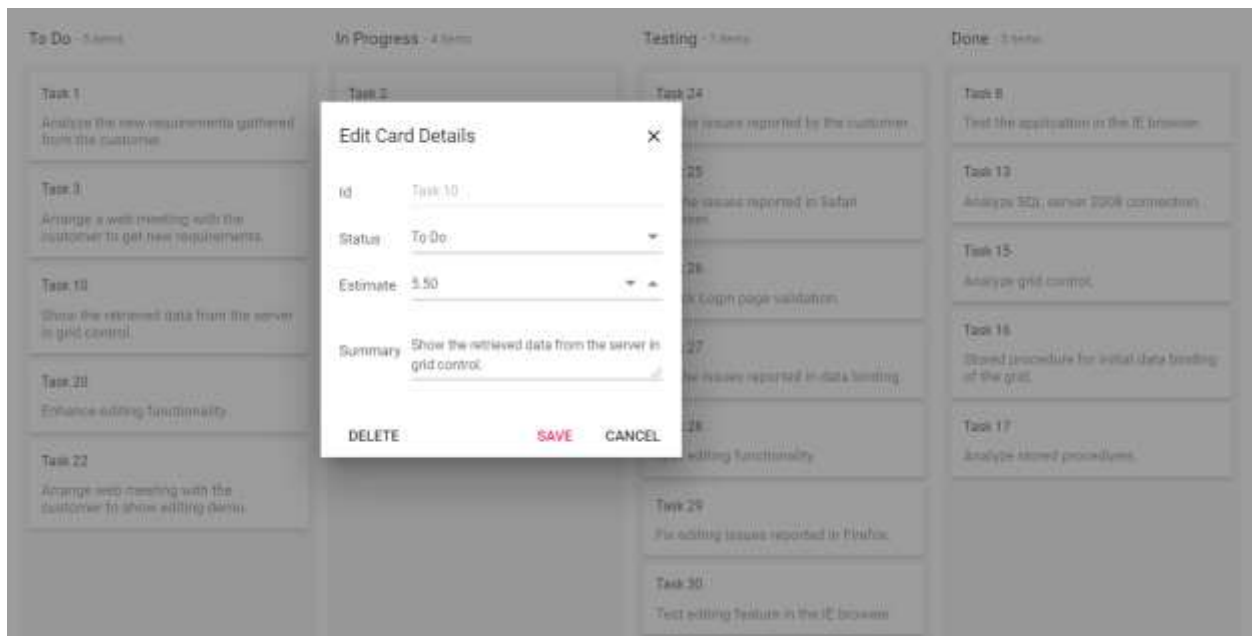
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.dialog = new KanbanDialogModels().DialogCards();
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Custom Fields label

By default, the fields **Key** mapping value is considered as a **Label** and you can change this label by using **Text** property.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DialogSettings((item) => item.Fields(ViewBag.dialog)).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    }
}
```



```

TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",

```



```

Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

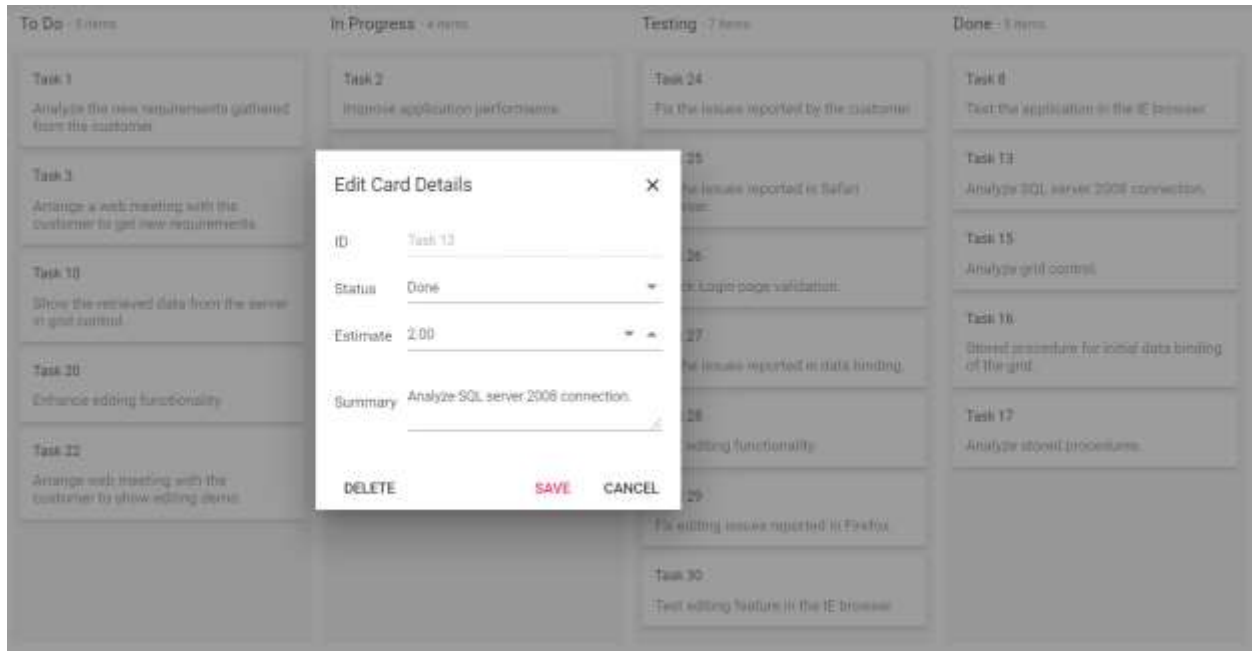
```
public class KanbanDialogModels
{
    public string text { get; set; }
    public string key { get; set; }
    public string type { get; set; }

    public List<KanbanDialogModels> DialogCards()
    {
        List<KanbanDialogModels> DialogCard = new
List<KanbanDialogModels>();
        DialogCard.Add(new KanbanDialogModels { text = "ID", key = "Id",
type = "Input" });
        DialogCard.Add(new KanbanDialogModels { key = "Status", type =
"DropDown" });
        DialogCard.Add(new KanbanDialogModels { key = "Estimate", type =
"Numeric" });
        DialogCard.Add(new KanbanDialogModels { key = "Summary", type =
"TextArea" });
        return DialogCard;
    }
}
```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.dialog = new KanbanDialogModels().DialogCards();
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.



Fields Validation

The dialog fields can be validated while click on the **Save** button. This can be achieved by using **ValidationRules** property.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DialogSettings((item) => item.Fields(ViewBag.dialog)).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {

```

```

        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation, Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database, SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database, Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",

```

```

Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

public class KanbanDialogModels
{
    public string text { get; set; }
    public string key { get; set; }
    public string type { get; set; }
    public object validationRules { get; set; }

    public List<KanbanDialogModels> DialogCards()
    {
        List<KanbanDialogModels> DialogCard = new
List<KanbanDialogModels>();
        DialogCard.Add(new KanbanDialogModels { key = "Id", type =
"Input" });
        DialogCard.Add(new KanbanDialogModels { key = "Status", type =
"DropDown" });
        DialogCard.Add(new KanbanDialogModels { key = "Estimate", type =
"Numeric" , validationRules = new { range =new int[] { 0, 1000 } } });
        DialogCard.Add(new KanbanDialogModels { key = "Summary", type =
"TextArea", validationRules = new { required = true } });
        return DialogCard;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {

```

```

        ViewBag.dialog = new KanbanDialogModels().DialogCards();
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Dialog Template

Using the dialog template, you can render your own dialog by defining the **Template** property. Initialize the template as SCRIPT element Id or HTML string which holds the template and map it to the template property.

CSHTML

```

<div class="col-lg-9 control-section">
    <div class="control_wrapper">
        <div class="kanban-section">
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
    {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id");
    }).DialogSettings(dialog => {
        dialog.Template("#dialogTemplate");
    }).DialogOpen("onDialogOpen").Created("onKanbanCreated").Render()
        </div>
    </div>
</div>
<div class="col-lg-3 property-section">
    <table id="property" title="Properties">
        <tr>
            <td>
                <button class="e-btn e-dialog-add" id="addNew">Add New
Card</button>
            </td>
        </tr>
    </table>
</div>
<script id='dialogTemplate' type="text/x-template">
    <table>
        <tbody>
            <tr>
                <td>
                    <td class="e-label">ID</td>
                    <td>
                        <input id="Id" name="Id" type="text" class="e-field"
value="{Id}" disabled required />
                    </td>
                </tr>
            <tr>
                <td class="e-label">Status</td>
                <td>

```

```

        <input type="text" name="Status" id="Status"
class="e-field" value=${Status} required />
    </td>
</tr>
<tr>
    <td class="e-label">Assignee</td>
    <td>
        <input type="text" name="Assignee" id="Assignee"
class="e-field" value=${Assignee} />
    </td>
</tr>
<tr>
    <td class="e-label">Priority</td>
    <td>
        <input type="text" name="Priority" id="Priority"
class="e-field" value=${Priority} />
    </td>
</tr>
<tr>
    <td class="e-label">Summary</td>
    <td>
        <textarea type="text" name="Summary" id="Summary"
class="e-field" value=${Summary}>${Summary}</textarea>
        <span class="e-float-line"></span>
    </td>
</tr>
</tbody>
</table>
</script>
<script>
    var kanbanObj;
    var statusData = ['Open', 'InProgress', 'Testing', 'Close'];
    var assigneeData = ['Nancy Davloio', 'Andrew Fuller', 'Janet
Leverling',
        'Steven walker', 'Robert King', 'Margaret hamilt', 'Michael
Suyama'];
    var priorityData = ['Low', 'Normal', 'Critical', 'Release Breaker',
'High'];
    function onKanbanCreated() {
        kanbanObj = this;
    }
    function onDialogOpen(args) {
        if (args.requestType !== 'Delete') {
            var curData = args.data;
            var filledTextBox = new ej.inputs.TextBox({});
            filledTextBox.appendTo(args.element.querySelector('#Id'));
            var numericObj = new ej.inputs.NumericTextBox({
                value: curData.Estimate, placeholder: 'Estimate',
            });
            numericObj.appendTo(args.element.querySelector('#Estimate'));
            var statusDropObj = new ej.dropdowns.DropDownList({
                value: curData.Status, popupHeight: '300px',
                dataSource: statusData, fields: { text: 'Status', value:
'Status' }, placeholder: 'Status'
            });
            statusDropObj.appendTo(args.element.querySelector('#Status'));
            var assigneeDropObj = new ej.dropdowns.DropDownList({

```



```

        value: curData.Assignee, popupHeight: '300px',
        dataSource: assigneeData, fields: { text: 'Assignee', value:
'Assignee' }, placeholder: 'Assignee'
    });
assigneeDropObj.appendTo(args.element.querySelector('#Assignee'));
    var priorityObj = new ej.dropdowns.DropDownList({
        value: curData.Priority, popupHeight: '300px',
        dataSource: priorityData, fields: { text: 'Priority', value:
'Priority' }, placeholder: 'Priority'
    });
    priorityObj.appendTo(args.element.querySelector('#Priority'));
    var textareaObj = new ej.inputs.TextBox({
        placeholder: 'Summary',
        multiline: true
    });
    textareaObj.appendTo(args.element.querySelector('#Summary'));
}
var count = 31;
document.getElementById('addNew').onclick = function () {
    var curData = { Id: 'Task ' + count, Status: 'Open', Priority:
'Normal', Assignee: 'Andrew Fuller', Estimate: 0, Tags: '', Summary: '' };
    kanbanObj.openDialog('Add', curData);
    count++;
}
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =

```

```

"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding

```

```

issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in

```

```

data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
return TaskDetails;
}
}

```

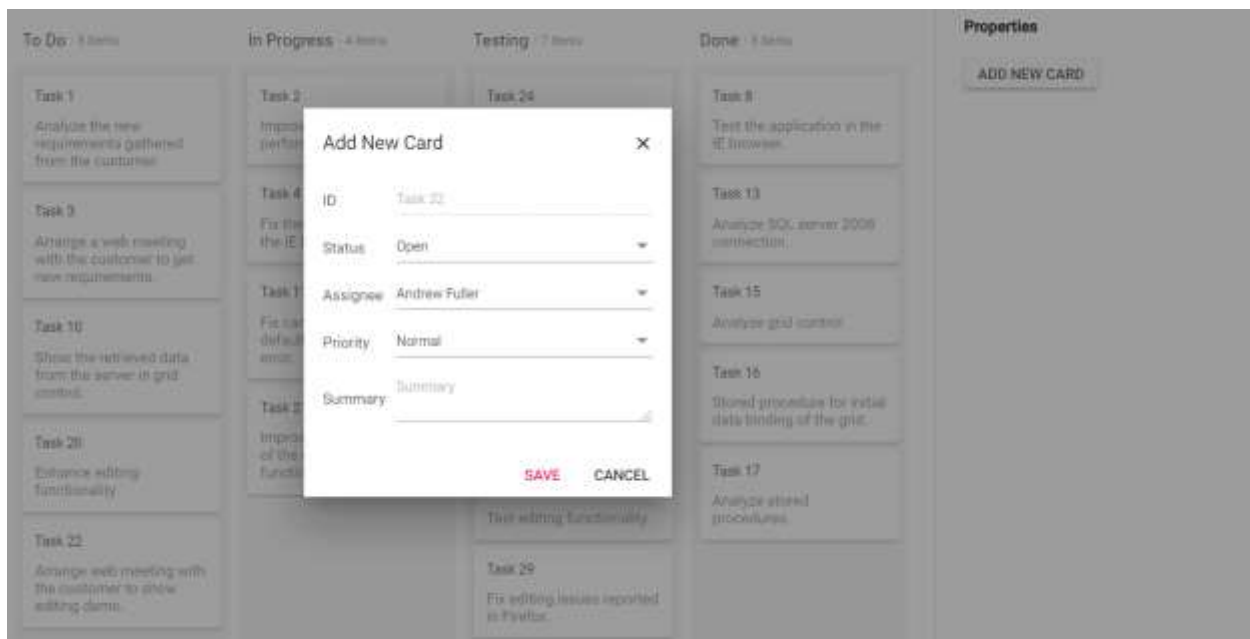
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        ViewBag.status = new KanbanDataModels().DialogStatus();
        ViewBag.assignee = new KanbanDataModels().AssigneeData();
        ViewBag.priority = new KanbanDataModels().PriorityData();
        return View();
    }
}

```

Output be like the below.



Prevent Dialog

The Kanban allows to prevent to open a dialog on card double-click by enabling `args.cancel` in `DialogOpen` event.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DialogOpen("onOpen").Render()

<script>
function dialogOpen(args: DialogEventArgs): void {
    args.cancel = true;
}
</script>
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    }
}
```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial

```

```

data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported

```



```

in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Persisting data in server

The modified card data can be persisted in the database using the RESTful web services. All the CRUD operations in the Kanban are done through **DataManager**. The **DataManager** has an option to bind all the CRUD related data in server-side.

Note: For your information, the ODataAdaptor persists data in the server as per OData protocol.

In the below section covers how to get the edited data details on the server-side using the **UrlAdaptor**.

URL adaptor

You can use the **UrlAdaptor** of **DataManager** when binding data source for remote data. In the initial load of Kanban, data are fetched from remote data and bound to the Kanban using **Url** property of **DataManager**.

You can map the CRUD operation in Kanban can be mapped to server-side controller actions using the properties **InsertUrl**, **RemoveUrl**, **UpdateUrl**, and **CrudUrl**.

- **InsertUrl** – You can perform single insertion operation on server-side.
- **UpdateUrl** – You can update single data on server-side.
- **RemoveUrl** – You can remove single data on server-side.
- **CrudUrl** – You can perform bulk data operation on server-side.

The following code example describes the above behavior.

CSHTML

```

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource(dataManger
=>
{
    dataManger.Url("DataSource").CrudUrl("UpdateData").Adaptor("UrlAdaptor").Cro
ssDomain(true);
}).Columns(col=> {

```



```

        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id");
    }).Render()

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}

```

The server-side controller code to handle the CRUD operations are as follows.

```

`typescript
private NORTHWNDEntities db = new NORTHWNDEntities();

public ActionResult DataSource() {
    var DataSource = db.Tasks.ToList();
    return Json(DataSource, JsonRequestBehavior.AllowGet);
}

public ActionResult Insert(Params value) {
    //Insert card data into the database
    return Json(value, JsonRequestBehavior.AllowGet);
}

public ActionResult Update(Params value) {
    //Update card data into the database
    return Json(value, JsonRequestBehavior.AllowGet);
}

public void Delete(Params value) {
    //Delete card data from the database
}

public class Params {
    public int Id { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
}

```

```
public string Assignee { get; set; }  
}  
`
```

Insert card

Using the `InsertUrl` property, you can specify the controller action mapping URL to perform insert operation on the server-side.

The following code example describes the above behavior.

```
`typescript  
public ActionResult Insert(Params value)  
{  
    //Insert card in the database  
}  
`
```

The newly added record details are bound to the `value` parameter.

Update card

Using the `UpdateUrl` property, the controller action mapping URL can be specified to perform save/update operation on the server-side.

The following code example describes the above behavior.

```
`typescript  
public ActionResult Update(Params value)  
{  
    //Update card data in the database  
}  
`
```

The updated record details are bound to the `value` parameter.

Delete card

Using the `RemoveUrl` property, the controller action mapping URL can be specified to perform delete operation on the server-side.

The following code example describes the above behavior.

```
`typescript  
public void Delete(int key)  
{  
    //Delete card in the database  
}  
`
```

The deleted card primary key value is bound to the `key` parameter.

CRUD URL

Using the `CrudUrl` property, the controller action mapping URL can be specified to perform all the CRUD operations at the server-side using a single method instead of specifying a separate controller action method for CRUD (insert, update and delete) operations.

The action parameter of `CrudUrl` is used to get the corresponding CRUD action.

The following code example describes the above behavior.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource(dataManger
=>
{
    dataManger.Url("DataSource").CrudUrl("UpdateData").Adaptor("UrlAdaptor").Cro
ssDomain(true);
}).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title = "Task
- 29001", Status = "Open", Summary = "Analyze the new requirements gathered
from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title = "Task
- 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title = "Task
- 29003", Status = "Open", Summary = "Arrange a web meeting with the
```

```

customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title = "Task
- 29004", Status = "InProgress", Summary = "Fix the issues reported in the
IE browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE",
Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title = "Task
- 29005", Status = "Review", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate =
3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title = "Task
- 29007", Status = "Validate", Summary = "Validate new requirements", Type =
"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title = "Task
- 29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title = "Task
- 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title = "Task
- 29011", Status = "Validate", Summary = "Validate the issues reported by
the customer.", Type = "Story", Priority = "High", Tags = "Validation,Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title = "Task
- 29015", Status = "Open", Summary = "Show the retrieved data from the
server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title = "Task
- 29016", Status = "InProgress", Summary = "Fix cannot open user's default
database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title = "Task
- 29017", Status = "Review", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title = "Task
- 29018", Status = "Close", Summary = "Analyze SQL server 2008 connection.",
Type = "Story", Priority = "Release Breaker", Tags = "Grid,Sql", Estimate =
2, Assignee = "Andrew Fuller", RankId = 4, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title = "Task
- 29019", Status = "Validate", Summary = "Validate databinding issues.",
Type = "Story", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title = "Task
- 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title = "Task
- 29021", Status = "Close", Summary = "Stored procedure for initial data
binding of the grid.", Type = "Others", Priority = "Release Breaker", Tags =
"Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId = 6, Color
= "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title = "Task
- 29022", Status = "Close", Summary = "Analyze stored procedures.", Type =
"Story", Priority = "Release Breaker", Tags = "Procedures", Estimate = 5.5,
Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title = "Task
- 29023", Status = "Validate", Summary = "Validate editing issues.", Type =
"Story", Priority = "Critical", Tags = "Editing", Estimate = 1, Assignee =
"Nancy Davloio", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title = "Task
- 29024", Status = "Review", Summary = "Test editing functionality.", Type =
"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title = "Task
- 29025", Status = "Open", Summary = "Enhance editing functionality.", Type
= "Improvement", Priority = "Low", Tags = "Editing", Estimate = 3.5,
Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title = "Task
- 29026", Status = "InProgress", Summary = "Improve the performance of the
editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title = "Task
- 29027", Status = "Open", Summary = "Arrange web meeting with the customer
to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title = "Task
- 29029", Status = "Review", Summary = "Fix the editing issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Editing,Fix",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6, Color = "#cc0000"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title = "Task
- 29030", Status = "Testing", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Critical", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title = "Task
- 29031", Status = "Testing", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title = "Task
- 29032", Status = "Testing", Summary = "Check Login page validation.", Type
= "Story", Priority = "Release Breaker", Tags = "Testing", Estimate = 0.5,
Assignee = "Michael Suyama", RankId = 3 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title = "Task
- 29033", Status = "Testing", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title = "Task
- 29034", Status = "Testing", Summary = "Test editing functionality.", Type
= "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title = "Task
- 29035", Status = "Testing", Summary = "Fix editing issues reported in
Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title = "Task
- 29036", Status = "Testing", Summary = "Test editing feature in the IE
browser.", Type = "Story", Priority = "Normal", Tags = "Testing", Estimate =
5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

`typescript

```

private NORTHWNDEntities db = new NORTHWNDEntities();

public ActionResult DataSource() {
    var DataSource = db.Tasks.ToList();
    return Json(DataSource, JsonRequestBehavior.AllowGet);
}

public ActionResult UpdateData(EditParams param) {
    if (param.action == "insert" || (param.action == "batch" && param.added != null)) {
        if (param.action == "insert") {
            db.Tasks.Add(param.value);
        } else {
            foreach (var temp in param.added) {
                db.Tasks.Add(temp);
            }
        }
    }

    if (param.action == "update" || (param.action == "batch" && param.changed != null)) {
        if (param.action == "update") {
            Task old = db.Tasks.Where(o => o.Id == param.value.Id).SingleOrDefault();

```

```
if (old != null) {
    db.Entry(old).CurrentValues.SetValues(param.value);
}
} else {
    foreach (var temp in param.changed) {
        Task old = db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault();
        if (old != null) {
            db.Entry(old).CurrentValues.SetValues(temp);
        }
    }
}

if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) {
    if (param.action == "remove") {
        int key = Convert.ToInt32(param.key);
        db.Tasks.Remove(db.Tasks.Where(o => o.Id == key).SingleOrDefault());
    } else {
        foreach (var temp in param.deleted) {
            db.Tasks.Remove(db.Tasks.Where(o => o.Id == temp.Id).SingleOrDefault());
        }
    }
}

db.SaveChanges();
return Json(param, JsonRequestBehavior.AllowGet);
}

public class EditParams {
    public string key { get; set; }
    public string action { get; set; }
    public List<Tasks> added { get; set; }
    public List<Tasks> changed { get; set; }
    public List<Tasks> deleted { get; set; }
    public Tasks value { get; set; }
}
```

Note: The `CrudUrl` is used to update the bulk data sent to the server-side. Multiple selections and `SortBy` as `Index` properties are used for `CrudUrl` properties to update the modified bulk data to the server-side.

Tooltip in ASP.NET MVC Kanban control

The tooltip is used to show the card information when the cursor hover over the card elements using the `EnableTooltip` property. Tooltip content is dynamically set based on hovering over the card elements.

Note: If you wish to show tooltip on Kanban board custom elements, you need to add `e-tooltip-text` class name of a particular element.

Tooltip template

You can customize the tooltip content with any HTML or CSS element and styling using the `TooltipTemplate` property. In the following demo, the tooltip is customized with HTML elements.

CSHTML

```
<div>
@Html.EJS().Kanban("kanban").KeyField("Status").EnableTooltip(true).TooltipTemplate("#tooltipTemp").DataSource((IEnumerable<object>) ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
</div>
<script id="tooltipTemp" type="text/x-template">
    <div class='e-kanbantooltiptemp'>
        <table>
            <tr>
                <td class="details">
                    <table>
                        <colgroup>
                            <col style="width:30%">
                            <col style="width:70%">
                        </colgroup>
                        <tbody>
                            <tr>
                                <td class="CardHeader">Assignee:</td>
                                <td>${Assignee}</td>
                            </tr>
                            <tr>
                                <td class="CardHeader">Type:</td>
                                <td>${Type}</td>
                            </tr>
                            <tr>
                                <td class="CardHeader">Estimate:</td>
                                <td>${Estimate}</td>
                            </tr>
                        </tbody>
                    </table>
                </td>
            </tr>
        </table>
    </div>
</script>
```



```
 Summary:</td>  <td>${Summary}</td> </tr> </tbody> </table> </td> </tr> </table> </div> </script> <style type="text/css"> .e-kanbantooltiptemp { width: 250px; padding: 3px; } .e-kanbantooltiptemp > table { width: 100%; } .e-kanbantooltiptemp td { vertical-align: top; } </style> | |
```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",

```

```

Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

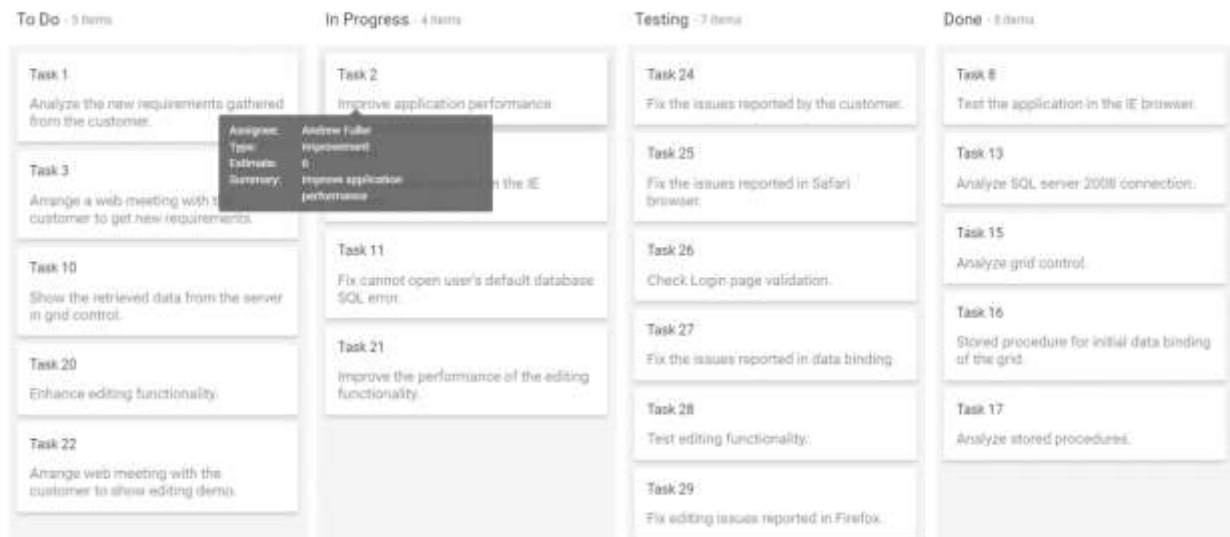
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Card Order in ASP.NET MVC Kanban control

By default, the Kanban cards are initially placed and drop the card inside the columns based on JSON data orders.

Cards are placed in a particular position in the columns when you drop the cards by specifying the **Priority** property, which is mapped from the datasource. This property allows the users to drop the cards in the Kanban board where exactly created on dropped clone. It is also helpful to render the cards based on the **Priority** property value.

The following cases are dynamically changed their **Priority** value when drop the cards.

- If the cell has no cards, the dropped card **Priority** value does not change.
- If the cell has one card and dropped a card to last position or previous/next cards that do not have continuous order, then the dropped card **Priority** value changed based on their previous card value.
- If the cell has one card and dropped a card on previous position, then compare both values and the dropped card **Priority** value is changed if the cards have continuous order otherwise not changed their value.
- When the previous and next cards does not have continuous order, the dropped card **Priority** value changed based on the previous card value.
- When previous and next cards have continuous order or odd/even value, then the dropped card followed by next all cards up to **continuous value** are dynamically changed their **Priority** value based on the **previous** card value.

For Example,

Continuous Order -

Column A having Card A with priority value 1, Card B with priority value 2 and Card C with priority value 3.

Column B having Card D with priority value 5. Dropped Card D between Card A and Card B. Now, Card D, B and C dynamically changed their priority value to 2, 3, 4.

Odd/Even order -

Column A having Card A with priority value 1, Card B with priority value 3 and Card C with priority value 5.

Column B having Card D with priority value 5. Dropped Card D between Card A and Card B. Now, Card D, B and C dynamically changed their priority value to 2, 3, 5.

Note: The **Priority** property mapping key value must be **number** format.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id").Priority("RankId");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
```

```

public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from

```

```

the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =

```



```

"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

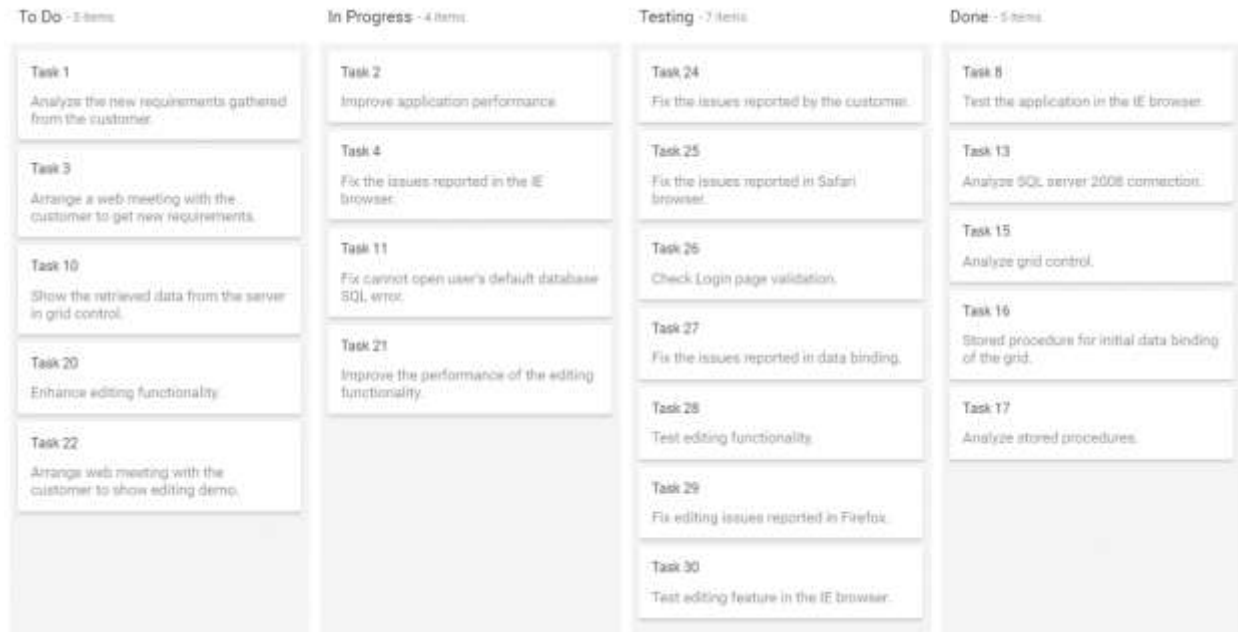
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Validation in ASP.NET MVC Kanban control

Validate particular column using the **MinCount** or **MaxCount** properties. The corresponding columns gets different appearance when validation fails. In default layout, **ConstraintType** property accept only **Column** type. In swimlane layout, accept both **Column** and **Swimlane** constraint type.

There are two types of constraints:

1. Column
2. Swimlane

Note: By default, the column count validation is performed based on Kanban **Columns**.

Minimum card limit

The **MinCount** property is used to specify the minimum cards hold on particular column or swimlane cell. If the column or swimlane total card count falls short of the minimum count value, it shows the column or cell background colour with validation fails.

Maximum card limit

The **MaxCount** property is used to specify the maximum cards hold on particular column or swimlane cell. If the column or swimlane cell total card count exceeds the maximum count value, it shows the column or cell background colour with validation fails.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col=> {
    col.HeaderText("To Do").KeyField("Open").ShowItemCount(true).MinCount(6).Add();
    col.HeaderText("In Progress").KeyField("InProgress").ShowItemCount(true).MaxCount(5).Add();
})
```

```
col.HeaderText("Testing").KeyField("Testing").ShowItemCount(true).MaxCount(5)
).MinCount(3).Add();
col.HeaderText("Done").KeyField("Close").ShowItemCount(true).Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    }
}
```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",

```

```

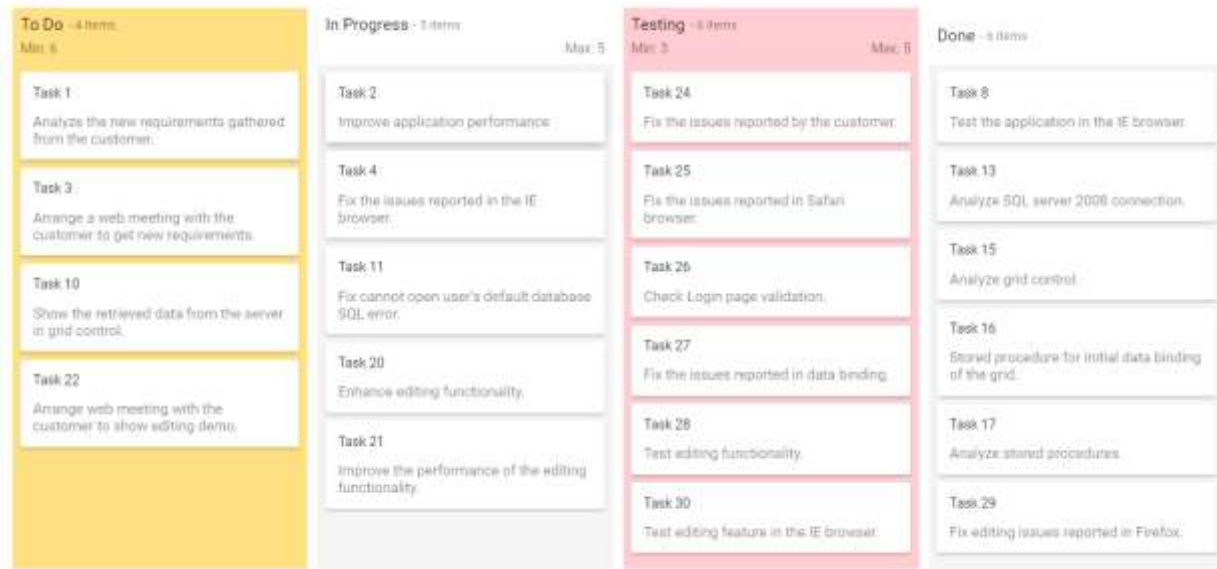
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.



Virtualization

Kanban allows you to load a large amount of data without any performance degradation. This feature can be enabled by setting the `EnableVirtualization` property in the Kanban to `true`.

Virtual scrolling

Virtual scrolling optimizes data rendering within each column when using large datasets. Only a subset of cards that are visible and about to be loaded on the screen are rendered. The number of records displayed in the Kanban is determined implicitly by the height of the Kanban area and the card height. The `CardHeight` property of Kanban can be used to set the card's height in pixel value. By default, the card height will be `auto`.

When the Kanban column is scrolled, the virtual scrolling feature dynamically loads additional data on demand into view and unloads the data that is no longer visible.

CSHTML

```
@Html.EJS().Kanban("KanbanVirtualScrolling").KeyField("Status").EnableTooltp(true).EnableVirtualization(true).DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Code Review").KeyField("Review").Add();
    col.HeaderText("Done").KeyField("Close").Add();
})
```

```

    }).CardSettings(card =>
    {

card.HeaderField("Id").SelectionType(SelectionType.Multiple).ContentField("Summary");
    }).DialogSettings((item) => item.Fields(ViewBag.dialogData)).Render()

```

DATASOURCE.CS

```

using System;
using System.Collections.Generic;
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public string StoryPoints { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",

```

```

Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

```



```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
    public List<KanbanDataModels> VirtualScrollKanbanData()
    {

```



```

        List<KanbanDataModels> KanbanData = new
List<KanbanDataModels>();
        string[] BUG_TASKS = { "UI component not displaying images in IE
browser", "Button not responding on hover action",
            "Text overlapping in mobile view",
            "Dropdown menu not functioning properly",
            "Form validation error",
            "Alignment issue in tables",
            "Column not loading completely",
            "Broken UI Designs",
            "Font size inconsistency",
            "UI element misaligned on scroll"
        };
        string[] FEATURE_TASKS = { "Implement new user registration
flow",
            "Add pagination to search results",
            "Improve accessibility for visually impaired users",
            "Create custom dashboard for users",
            "Develop user profile editing functionality",
            "Integrate with third-party API for weather data",
            "Implement social media sharing for articles",
            "Add support for multiple languages",
            "Create onboarding tutorial for new users",
            "Implement push notifications for mobile app"
        };
        string[] EPIC_TASKS = { "Revamp UI design for entire
application",
            "Develop mobile application for iOS and Android",
            "Create API for integration with external systems",
            "Implement machine learning algorithms for personalized
recommendations",
            "Upgrade database infrastructure for scalability",
            "Integrate with payment gateway for subscription model",
            "Develop chatbot for customer support",
            "Implement real-time collaboration features for team
projects",
            "Create analytics dashboard for administrators",
            "Introduce gamification elements to increase user
engagement",
        };
        string[] assignee = { "Andrew Fuller", "Janet Leverling",
"Steven walker", "Robert King", "Margaret hamilt", "Nancy Davloio",
"Margaret Buchanan", "Laura Bergs", "Anton Fleet", "Jack Kathryn", "Martin
Davolio", "Fleet Jack" };
        string[] status = { "Open", "InProgress", "Review", "Close",
"Testing" };
        string[] priority = { "Ultra-Critical", "Critical", "High",
"Normal", "Low" };
        string[] types = { "Epic", "Bug", "Story" };
        string[] tagsField = { "Feature", "Bug", "Enhancement",
"Documentation", "Automation", "Mobile", "Web", "iOS", "Safari", "Chrome",
"Firefox", "Manual Testing" };
        string[] storyPoints = { "1", "2", "3", "3.5", "4", "4.5", "5",
"6", "7.5", "8" };
        int count = 600000;
        int id = 500000;
        Random rnd = new Random();

```

```

        int month = rnd.Next(1, 13);
        for (int a = 500000; a < count; a++) {
            string typeValue = types[rnd.Next(0, 2)];
            string summary = typeValue == "Bug" ? BUG_TASKS[rnd.Next(0,
8)] : typeValue == "Story" ? FEATURE_TASKS[rnd.Next(0, 9)] :
EPIC_TASKS[rnd.Next(0, 9)];
            KanbanData.Add(new KanbanDataModels {
                Id = id.ToString(),
                Type = typeValue,
                Priority = priority[rnd.Next(0, 4)],
                Status = status[rnd.Next(0, 4)],
                Assignee = assignee[rnd.Next(0, 11)],
                StoryPoints = storyPoints[rnd.Next(0, 8)],
                Tags = tagsField[rnd.Next(0, 11)] + "," +
tagsField[rnd.Next(0, 11)],
                Title = "Task " + id,
                Summary = summary,
            });
            id++;
        }
        return KanbanData;
    }
}

public class KanbanDialogModels
{
    public string text { get; set; }
    public string key { get; set; }
    public string type { get; set; }
    public List<KanbanDialogModels> VirtualScrollDialogCardField()
    {
        List<KanbanDialogModels> VirtualScrollDialogCardField = new
List<KanbanDialogModels>();
        VirtualScrollDialogCardField.Add(new KanbanDialogModels { text =
"ID", key = "Id", type = "TextBox" });
        VirtualScrollDialogCardField.Add(new KanbanDialogModels { text =
"Status", key = "Status", type = "DropDown" });
        VirtualScrollDialogCardField.Add(new KanbanDialogModels { key =
"StoryPoints", text = "Story Points", type = "TextBox" });
        VirtualScrollDialogCardField.Add(new KanbanDialogModels { key =
"Summary", text = "Summary", type = "TextArea" });
        return VirtualScrollDialogCardField;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().VirtualScrollKanbanData();
        ViewBag.dialogData = new
KanbanDialogModels().VirtualScrollDialogCardField();
        return View();
    }
}

```

Configure the remote data service

When the remote data is configured for the [DataSource](#), the service method will receive an additional `KanbanVirtualization` parameter to handle the initial data load for Kanban Virtualization.

To handle Kanban virtual scrolling, the server-side code needs to handle the `Where` and `Take` queries differently using the `KanbanVirtualization` parameter. The following is the example code for handling Kanban virtualization's initial data load using the `KanbanVirtualization` parameter.

HOMECONTROLLER.CS

```
public IActionResult LoadCard([FromBody] ExtendedDataManagerRequest dm)
{
    kanbanData = _context.KanbanCards.ToList();
    IEnumerable<KanbanCard> DataSource = kanbanData.AsEnumerable();
    DataOperations operation = new DataOperations();
    // For normal kanban data load `Where` query handling.
    if (dm.Where != null && dm.Where.Count > 0 && dm.KanbanVirtualization !=
        "KanbanVirtualization")
    {
        dm.Where[0].value = dm.Where[0].value.ToString();
        DataSource = operation.PerformFiltering(DataSource, dm.Where,
            dm.Where[0].Operator);
    }
    if (dm.Skip != 0)
    {
        DataSource = operation.PerformSkip(DataSource, dm.Skip);
    }
    // For normal Kanban data load `Take` query handling.
    if (dm.Take != 0 && dm.KanbanVirtualization != "KanbanVirtualization")
    {
        DataSource = operation.PerformTake(DataSource, dm.Take);
    }
    // For Kanban virtual scrolling data load `Where` and `Take` query handling.
    var columnCount = new List<KeyValuePair<string, int>>();
    if (dm.KanbanVirtualization == "KanbanVirtualization" && dm.Where != null &&
        dm.Where.Count > 0 && dm.Take != 0)
    {
        IEnumerable<KanbanCard> currentData = new List<KanbanCard>();
        List<WhereFilter> currentFilter = new List<WhereFilter>();
        for (int i = 0; i < dm.Where.Count; i++)
        {
            dm.Where[i].value = dm.Where[i].value.ToString();
            currentFilter.Add(dm.Where[i]);
            var filterData = operation.PerformFiltering(DataSource, currentFilter,
                dm.Where[i].Operator);
            columnCount.Add(new KeyValuePair<string, int>(dm.Where[i].value.ToString(),
                filterData.Count()));
            filterData = operation.PerformTake(filterData, dm.Take);
            currentData = currentData.Concat(filterData);
            currentFilter.Clear();
        }
        DataSource = currentData;
    }
    // To return the data for Kanban virtual scrolling.
    if (dm.KanbanVirtualization == "KanbanVirtualization") {
```

```
return Json(new { result = DataSource, count = columnCount });
}
// To return the data for Kanban virtual scrolling.
else
{
return Json(DataSource);
}
}
```

Limitations for virtual scrolling

- When virtualization is enabled in a Kanban board and the card height is not explicitly set, it will not default to `auto` height. Instead, a fixed height of `100px` will be applied to the cards. It's important to note that the card height should be specified in pixel values, as percentage values are not accepted.
- When a card is dragged and dropped, the index position of the card will not be preserved when scrolling through the column.
- Virtualization is not supported for swimlanes in the Kanban board.

Globalization in ASP.NET MVC Kanban control

The localization library allows you to localize the default text content of the Kanban to different cultures using the `Locale` property.

In Kanban, total count and min or max count text alone will be localized based on culture.

| | | |
|---------------|--|--|
| Locale key | en-US (default) | |
| ----- | ----- | |
| items | items | |
| min | Min | |
| max | Max | |
| cardsSelected | Cards Selected | |
| addTitle | Add New Card | |
| editTitle | Edit Card Details | |
| deleteTitle | Delete Card | |
| deleteContent | Are you sure you want to delete this card? | |
| save | Save | |
| delete | Delete | |
| cancel | Cancel | |
| yes | Yes | |
| no | No | |
| close | Close | |
| noCard | No cards to display | |

| unassigned | Unassigned |

Loading translations

To load translation object in an application, use `load` function of `L10n` class.

The following example demonstrates the Kanban in `Deutsch` culture.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)
ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Locale("de").SwimlaneSettings(swim => {
    swim.KeyField("Assignee");
}).Render()
<script>
    ej.base.L10n.load({
        'de': {
            'kanban': {
                'items': 'Artikel',
                'min': 'Min',
                'max': 'Max',
                'cardsSelected': 'Karten ausgewählt',
                'addTitle': 'Neue Karte hinzufügen',
                'editTitle': 'Kartendetails bearbeiten',
                'deleteTitle': 'Karte löschen',
                'deleteContent': 'Möchten Sie diese Karte wirklich
löschen?',
                'save': 'speichern',
                'delete': 'Löschen',
                'cancel': 'Stornieren',
                'yes': 'Ja',
                'no': 'Nein',
                'close': 'Schließen',
                'noCard': 'Keine Karten zum Anzeigen',
                'unassigned': 'nicht zugewiesen'
            }
        }
    })
</script>
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
```

```

public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation, Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database, SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
}

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
        "Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
        default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
        "Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
        4, Color = "#cc0000" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
        "Task - 29017", Status = "Review", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
        "Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
        connection.", Type = "Story", Priority = "Release Breaker", Tags =
        "Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
        "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
        "Task - 29019", Status = "Validate", Summary = "Validate databinding
        issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
        1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
        "Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
        "Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
        "Margaret hamilt", RankId = 5, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
        "Task - 29021", Status = "Close", Summary = "Stored procedure for initial
        data binding of the grid.", Type = "Others", Priority = "Release Breaker",
        Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
        6, Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
        "Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
        Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
        = 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
        "Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
        Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
        Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
        "Task - 29024", Status = "Review", Summary = "Test editing functionality.",
        Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
        Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues

```

```

reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

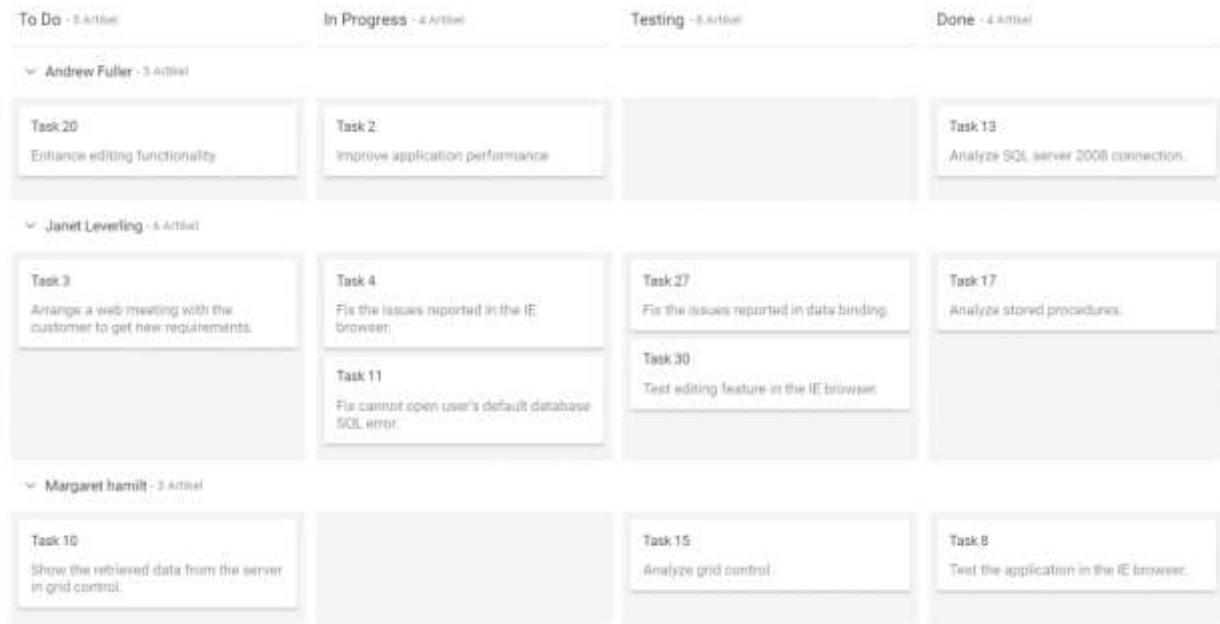
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Right to left (RTL)

The Kanban provides an option to switch its text direction and layout from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable right-to-left mode in Kanban, set the `EnableRtl` to true.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").EnableRtl(true).DataSource((
IEnumerable<object>) ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim => {
    swim.KeyField("Assignee");
}).Locale("ar").Render()
<script>
ej.base.L10n.load({
    'ar': {
        'kanban': {
            'items': 'العناصر',
            'min': 'أنا',
            'max': 'ماكس',
            'cardsSelected': 'تم تحديد البطاقات',
            'addTitle': 'إضافة بطاقة جديدة',
            'editTitle': 'تحرير تفاصيل البطاقة',
            'deleteTitle': 'حذف البطاقة',
            'deleteContent': 'هل أنت متأكد أنك تريد حذف هذه البطاقة؟',
            'save': 'حفظ',
            'delete': 'حذف',
            'cancel': 'إلغاء',
```

```

        'yes': 'نعم',
        'no': 'لا',
        'close': 'قريب'
        'noCard': 'لا توجد بطاقات لعرضها',
        'unassigned': 'غير معين'
    }
}
})
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",

```

```

Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

```

```

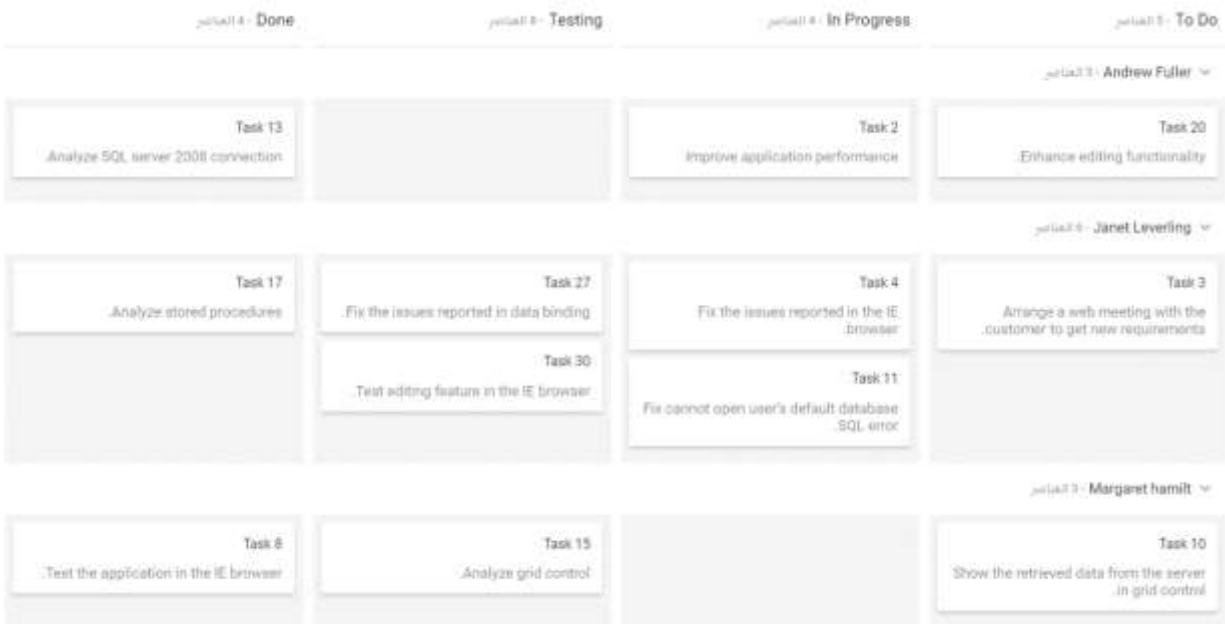
        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
        "Task - 29024", Status = "Review", Summary = "Test editing functionality.",
        Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
        Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
        customer to show editing demo.", Type = "Others", Priority = "High", Tags =
        "Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
        Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.



Kanban dimensions in ASP.NET MVC Kanban control

The Kanban dimensions refers to both height and width of the entire layout and it accepts three types of values.

- Auto
- Pixel
- Percentage

Auto height and width

When height and width of the Kanban are set to **auto**, it will try as hard as possible to keep an element the same width as its parent container. In other words, the parent container that holds Kanban, its width or height will be the sum of its children. By default, Kanban is assigned with **auto** values for both the height and width properties.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").Height("auto").Width("auto")
.DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
```

```

        {
            col.HeaderText("To Do").KeyField("Open").Add();
            col.HeaderText("In Progress").KeyField("InProgress").Add();
            col.HeaderText("Testing").KeyField("Testing").Add();
            col.HeaderText("Done").KeyField("Close").Add();
        }).CardSettings(card => {
            card.ContentField("Summary").HeaderField("Id");
        }).Render()
    }
}

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",

```



```

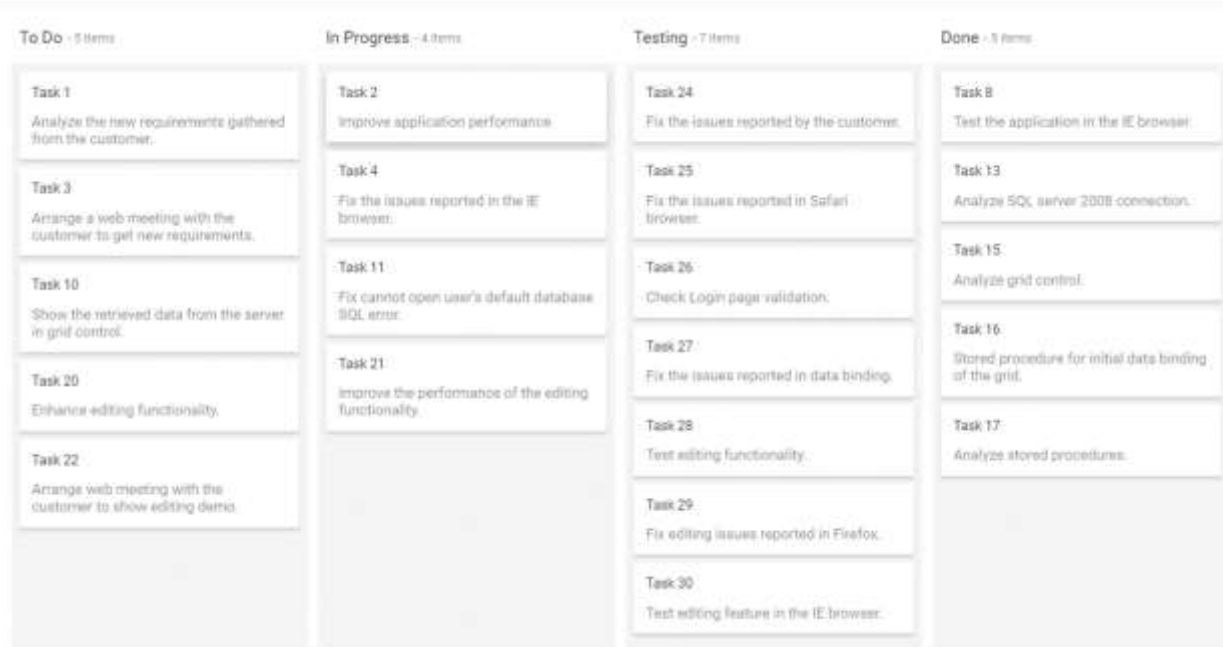
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS


```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}
```

Output be like the below.



Height and width in pixel

The Kanban height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").Height("550").Width("650px")
.DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}) .CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}) .Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
}
```

```

public string Id { get; set; }
public string Title { get; set; }
public string Status { get; set; }
public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
        "Task - 29015", Status = "Open", Summary = "Show the retrieved data from
        the server in grid control.", Type = "Story", Priority = "High", Tags =
        "Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
        Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
        "Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
        default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
        "Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
        4, Color = "#cc0000" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
        "Task - 29017", Status = "Review", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
        "Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
        connection.", Type = "Story", Priority = "Release Breaker", Tags =
        "Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
        "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
        "Task - 29019", Status = "Validate", Summary = "Validate databinding
        issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
        1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
        "Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
        "Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
        "Margaret hamilt", RankId = 5, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
        "Task - 29021", Status = "Close", Summary = "Stored procedure for initial
        data binding of the grid.", Type = "Others", Priority = "Release Breaker",
        Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
        6, Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
        "Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
        Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
        = 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
        "Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
        Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
        Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
        "Task - 29024", Status = "Review", Summary = "Test editing functionality.",
        Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
        Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
        "Task - 29025", Status = "Open", Summary = "Enhance editing
        functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
        Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
        });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
        "Task - 29026", Status = "InProgress", Summary = "Improve the performance
        of the editing functionality.", Type = "Epic", Priority = "High", Tags =
        "Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
        "#6d7492" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
        "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the

```

```

customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

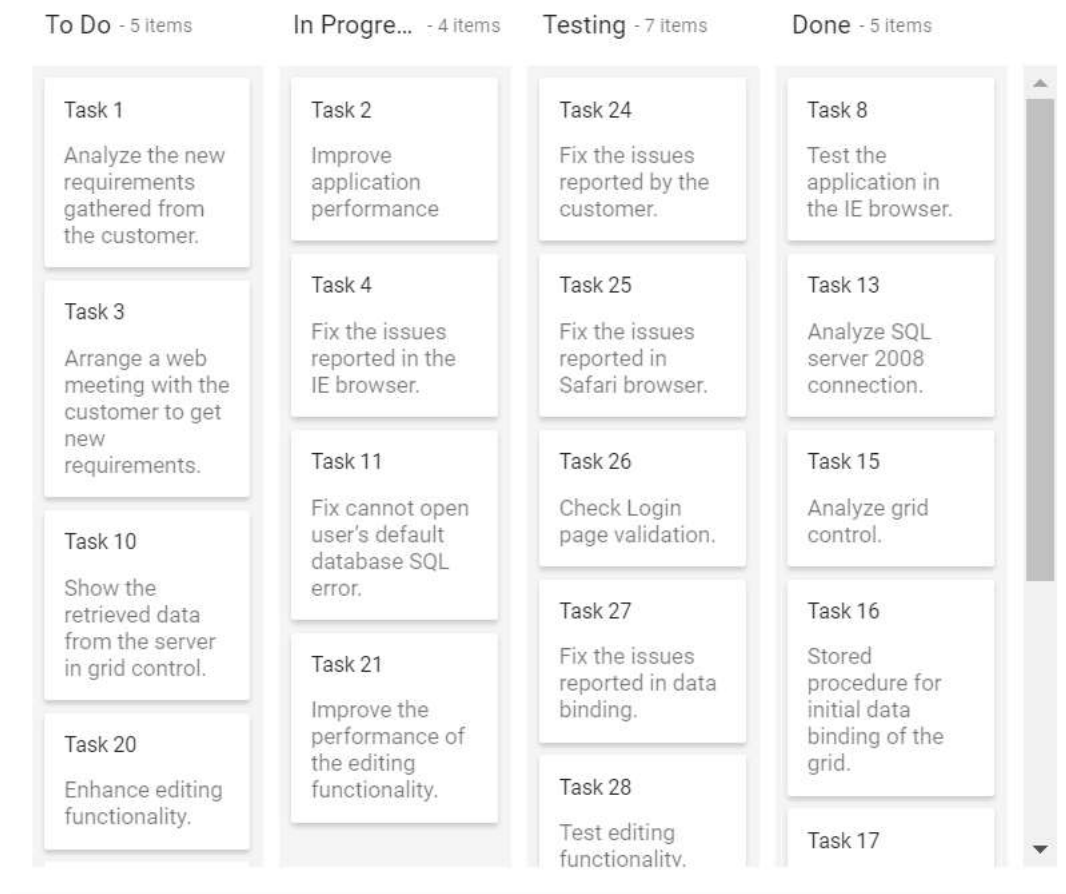
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



Height and width in percentage

When height and width of the Kanban are given in percentage, it will make the Kanban as wide as the parent container.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").Height("100%").Width("100%")
.DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
}
```

```

public string Summary { get; set; }
public string Type { get; set; }
public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation, Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =

```

```

"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

```



```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
        "Task - 29029", Status = "Review", Summary = "Fix the editing issues
        reported by the customer.", Type = "Bug", Priority = "Low", Tags =
        "Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
        Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
        "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
        the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
        Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
        "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
        Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
        "Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
        "Task - 29032", Status = "Testing", Summary = "Check Login page
        validation.", Type = "Story", Priority = "Release Breaker", Tags =
        "Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
        "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
        data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
        Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
        "Task - 29034", Status = "Testing", Summary = "Test editing
        functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
        Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
        "Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
        in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
        Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
        "Task - 29036", Status = "Testing", Summary = "Test editing feature in the
        IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
        Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

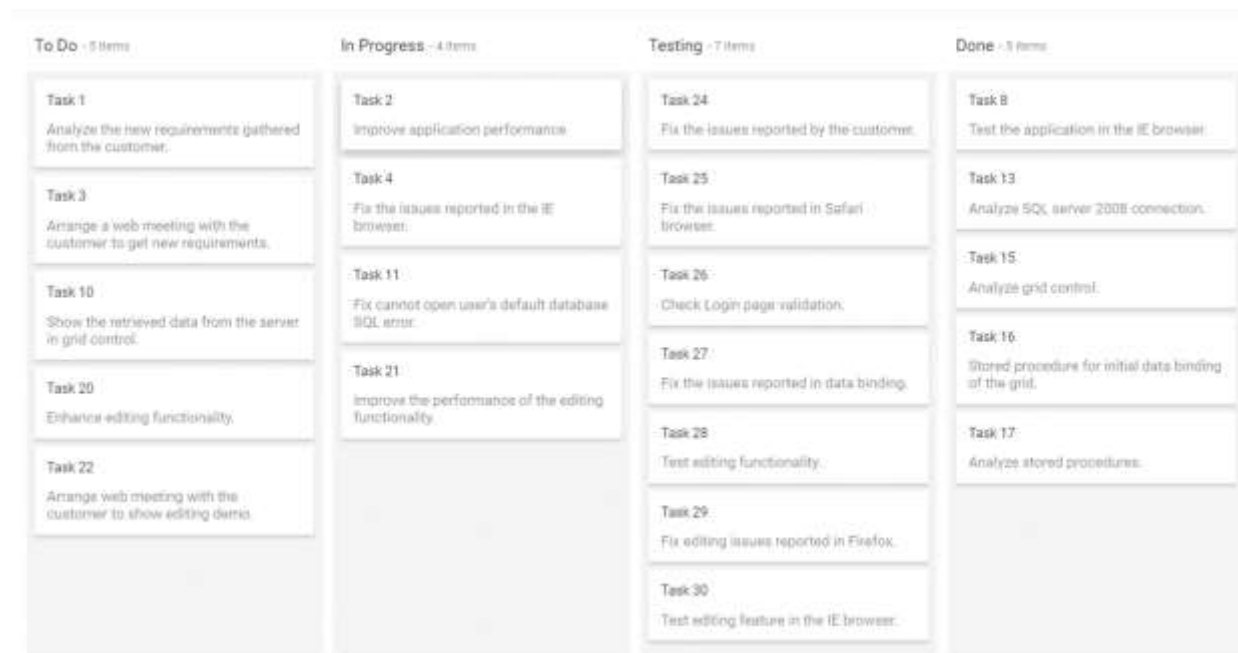
CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Output be like the below.



State Persistence in ASP.NET MVC Kanban control

State persistence refers to the Kanban state maintained in the browser's [LocalStorage](#) even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores Kanban datasource, column or swimlane expand/collapse state in the local storage when the `EnablePersistence` is defined as true.

CSHTML

```
@Html.EJS().Kanban("kanban").EnablePersistence(true).KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
```

```

public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =

```

```

"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
        return TaskDetails;
    }
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Responsive mode

The Kanban component has support for responsive behavior based on the client browser's width and height.

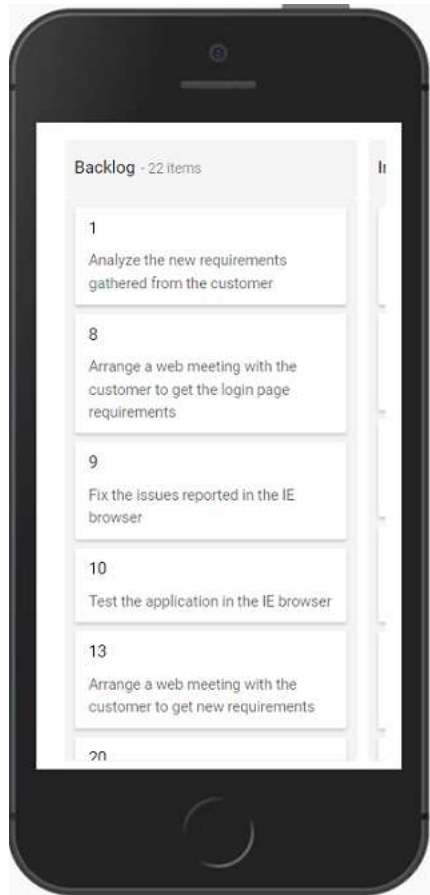
Layouts

Possible layouts are:

- Default Layout
- Swimlane Layout

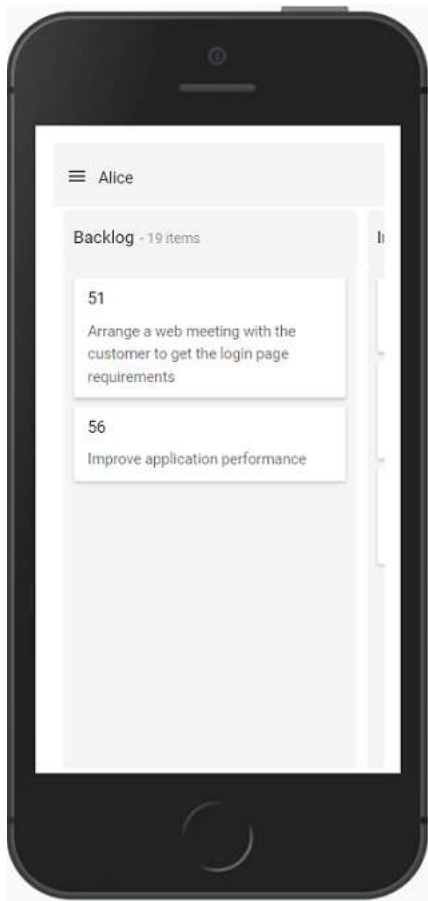
Default Layout

Kanban user interface is customized and redesigned for the best view on small screens. In responsive mode, the first column occupies 80% and the second column occupies 20% of the screen layout. Tap and hold the Kanban card to drag and drop it. Swipe left or right to view the columns.



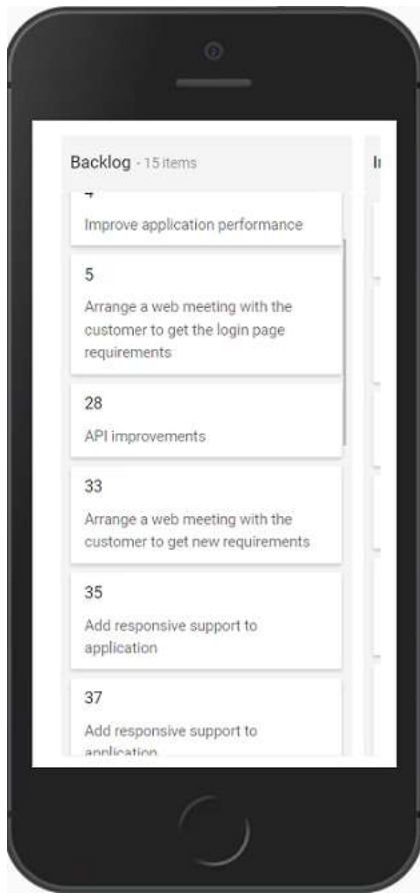
Swimlane Layout

Kanban swimlane header is rendered with menu icon on top of the kanban board. It will show all the available swimlane groups of the header text with a popup when clicking the menu icon. Swimlane selected grouped header text resultant data is rendered on the Kanban board. By default, the first swimlane grouped header text is selected and the resultant data is shown on the Kanban board. The Kanban board data will be changed when changing the swimlane group header text.



Scrolling

Column scrolling will be shown when exceeding the screen size in the columns.

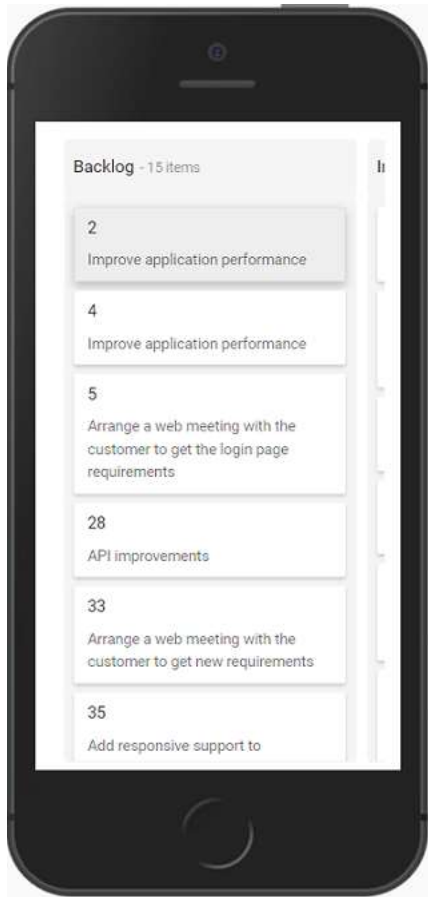


Selection

Select particular cards in the Kanban board by tapping the card.

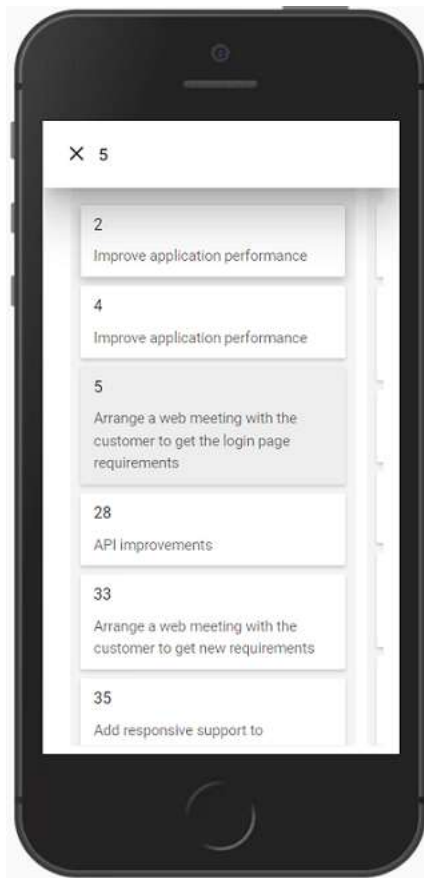
Single Selection

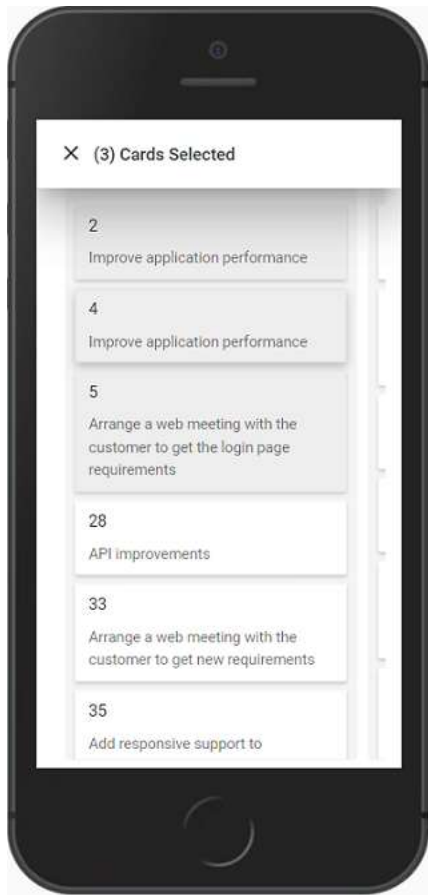
Single card will be selected when you tap the card once and selection will be removed when you select another card.



Multiple Selection

Enable [selectionType](#) as **Multiple** to select multiple cards. It will open the popup on the screen top. Selected card header text will be shown when selecting single card with a tap and hold action. If single card is selected, only tap action is required to select multiple cards. Multiple Selected card count will be shown on the popup when selecting multiple cards.





Styling and appearance

To modify the Kanban appearance, you need to override the default CSS of Kanban. Also, you have an option to create your own custom theme using our [Theme Studio](#). Find the list of CSS classes in Kanban.

| Css class | Purpose |
|---|------------------|
| ----- | ----- |
| .e-kanban | .e-kanban-table |
| customize the kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| Header cells of kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-header-wrap | |
| .e-header-title | |
| Header title of kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-min-color | |
| Header cells minimum color of kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-max-color | |
| Header cells maximum color of kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-collapsed | |
| .e-min-color | |
| Header cells of collapsed column minimum color in column constraint type of kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-collapsed | |
| .e-max-color | |
| Header cells of collapsed column maximum color in column constraint type of kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-header-text | |
| Header text of Kanban. | |
| .e-kanban | .e-kanban-header |
| .e-header-cells | |
| .e-item-count | |
| Header cells Item count of Kanban. | |

| .e-kanban .e-kanban-header .e-header-cells .e-limits | Header cells limits in column constraint type of kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits .e-min-count | Header cells minimum count of kanban. |

| .e-kanban .e-kanban-header .e-header-cells .e-limits .e-max-count | Header cells maximum count of kanban. |

| .e-kanban .e-kanban-content | Customize kanban Content. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits | Content cells limits in swimlane constraint type of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits .e-min-count | Content cells minimum count of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-limits .e-max-count | Content cells maximum count of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-min-color | Content cells minimum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-max-color | Content cells maximum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-collapsed .e-collapse-header-text | Content cells of collapsed header text. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells.e-collapsed .e-collapse-header-text .e-item-count | Content cells of collapsed header text Item count. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-show-add-button | Add button in content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-show-add-button .e-show-add-icon | Customize content cells add icon of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-empty-card | Empty content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card | Customize cards in kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-header .e-card-header-title | Cards header title of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-footer | Cards footer of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-content | Cards content of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card.e-card-color | Cards color of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-tags | Customize Card tags of kanban. |

| .e-kanban .e-kanban-content .e-content-row .e-content-cells .e-card-wrapper .e-card .e-card-tag |
Card tag of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-row-expand | Content cells of swimlane row expand of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-row-collapse | Content cells of swimlane row collapse of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-swimlane-text | Content cells of swimlane header text of kanban. |

| .e-kanban .e-kanban-content .e-content-row.e-swimlane-row .e-content-cells .e-swimlane-header .e-item-count | Content cells of swimlane items count of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells | swimlane content cells of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-dropping |
Customize swimlane content cells card dropping of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells .e-card-wrapper |
Swimlane content cells of card wrapper. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-min-color |
Swimlane content cells of minimum color of kanban. |

| .e-kanban .e-kanban-content .e-content-row:not(.e-swimlane-row) .e-content-cells.e-max-color |
Swimlane content cells of maximum color of kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-header-text | Header text of Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-item-count | Header cells Item count of Kanban. |

| .e-kanban .e-kanban-table .e-header-cells .e-column-expand | Header cells of toggle icon in column expand. |

| .e-kanban .e-kanban-table .e-header-cells .e-column-collapse | Header cells of toggle icon in column collapse. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row:not(.e-swimlane-row) .e-content-cells |
swimlane content cells of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row.e-swimlane-row .e-swimlane-text | Content cells of swimlane header text of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row.e-swimlane-row .e-item-count | Content cells of swimlane items count of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-content-row .e-show-add-button .e-show-add-icon | Add icon in content cells of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card.e-selection | Selected card of kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-header | Cards header in kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-content | Cards content in kanban. |

| .e-kanban .e-kanban-table.e-content-table .e-card .e-card-tag.e-card-label | Cards label in kanban. |

To set fixed position to the Kanban header

The Fixed header in Kanban control can be customized in following ways,

By setting a fixed height to the Kanban content,

```
`CSS
.e-kanban .e-kanban-content {
height: 500px;
}
`
```

By customizing the CSS for the Kanban header.

```
`CSS
.e-kanban-header {
position: -webkit-sticky;
position: sticky;
z-index: 100;
top: 0;
}
`
```

Note: It will not affect the Kanban content's height.

Accessibility in ASP.NET MVC Kanban control

The Kanban control has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties. This control is characterized by complete ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The accessibility compliance for the Kanban control is outlined below.

| | | |
|-------------------------------------|---|--|
| Accessibility Criteria | Compatibility | |
| -- | -- | |
| WCAG 2.2 Support | ! [Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) | |
| Section 508 Support | ! [Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) | |
| Screen Reader Support | ! [Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) | |
| Right-To-Left Support | ! [Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) | |
| Color Contrast | ! [Intermediate](https://cdn.syncfusion.com/content/images/documentation/partial.png) | |
| Mobile Device Support | ! [Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) | |

| [Keyboard Navigation Support](#) |

![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

| Accessibility Checker Validation |

![Intermediate](https://cdn.syncfusion.com/content/images/documentation/partial.png) |

| Axe-core Accessibility Validation |

![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

![Yes](https://cdn.syncfusion.com/content/images/documentation/full.png) - All features of the control meet the requirement.

![Intermediate](https://cdn.syncfusion.com/content/images/documentation/partial.png) - Some features of the control do not meet the requirement.

![No](https://cdn.syncfusion.com/content/images/documentation/not-supported.png) - The control does not meet the requirement.

WAI-ARIA attributes

The Kanban control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Kanban control:

| Attributes | Purpose |

| --- | --- |

| **aria-label** | It helps to provides information about elements in a kanban control for assistive technology. |

| **aria-expanded** | Attributes indicate the state of a collapsible element. |

| **aria-selected** | This attribute is assigned to the Kanban control for the selection of elements, and its default value is **false**. The value changes to true when the user selects a Kanban card. |

| **aria-grabbed** | Indicates whether the attribute is set to true. It has been selected for dragging. If this attribute is set to false, the element can be grabbed for a drag-and-drop operation but will not be currently grabbed. |

| **aria-describedby** | This attribute contains the ID of the Kanban header column to indicate that the attribute establishes an association between the Kanban header column and the Kanban column body. |

| **aria-roledescription** | This attribute is assigned to the Kanban control and is used to provide alternative descriptions for card elements. |

Keyboard interaction

The Kanban control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Kanban control.

| **Press** | **To do this** |

| --- | --- |

| **Home** | To select the first card in the kanban |

| **End** | To select the last card in the kanban |

| **Arrow Up** | Select the card through the up arrow |

| **Arrow Down** | Select the card through the down arrow |

| **Arrow Right** | Move the column selection to the right |

| **Arrow Left** | Move the column selection to the left |

| **Ctrl + Enter** | Used to select the multi cards |

| **Ctrl + Space** | Used to select the multi cards |

| **Shift + Arrow Up** | Used to select the multiple cards towards up |

| **Shift + Arrow Down** | Used to select the multiple cards towards down |

| **Shift + Tab** | Reverse order of the tab action |

| **Enter** | Open the selected cards |

| **Tab** | To navigate the Kanban column |

| **Delete** | To delete the selected cards |

| **ESC** | Escape from the modified details |

| **Space** | Used to open the card edit dialog based on the column selection |

Disable keyboard interaction

Disables all the functionalities in the Kanban board performed using keyboard by setting the `AllowKeyboard` properties to `False`.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).AllowKeyboard(false).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card =>
{
    card.ContentField("Summary").HeaderField("Id");
}).Render()
```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze,Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    }
}

```



```

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance

```

```

of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

CONTROLLER.CS

```

public class HomeController : Controller
{
    public ActionResult Index()
    {
        ViewBag.data = new KanbanDataModels().KanbanTasks();
        return View();
    }
}

```

Ensuring accessibility

The Kanban control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Kanban control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Kanban control with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC control](#)

How To

Add header double click

You can bind the header double click event by using the [dataBound](#) event at the initial rendering. You can get the column header text when you double click on the headers.

CSHTML

```
@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).DataBound("onDataBound").Render()

<script>
function onDataBound(args) {
    var headerEle = document.querySelector('.e-header-row');
    headerEle.addEventListener("dblclick", function (e) {
        var target = ej.base.closest(e.target, '.e-header-cells');
        ej.popups.DialogUtility.alert({
            title: 'Header',
            content: "Double clicked on " + target.querySelector('.e-header-text').innerText + " header",
            showCloseIcon: true,
            closeOnEscape: true,
            animationSettings: { effect: 'Zoom' }
        });
    });
}
</script>
```

DATASOURCE.CS

```
public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
```

```

public string Priority { get; set; }
public string Tags { get; set; }
public Double Estimate { get; set; }
public string Assignee { get; set; }
public int RankId { get; set; }
public string Color { get; set; }
public List<KanbanDataModels> KanbanTasks()
{
    List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
    TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation, Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database, SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues

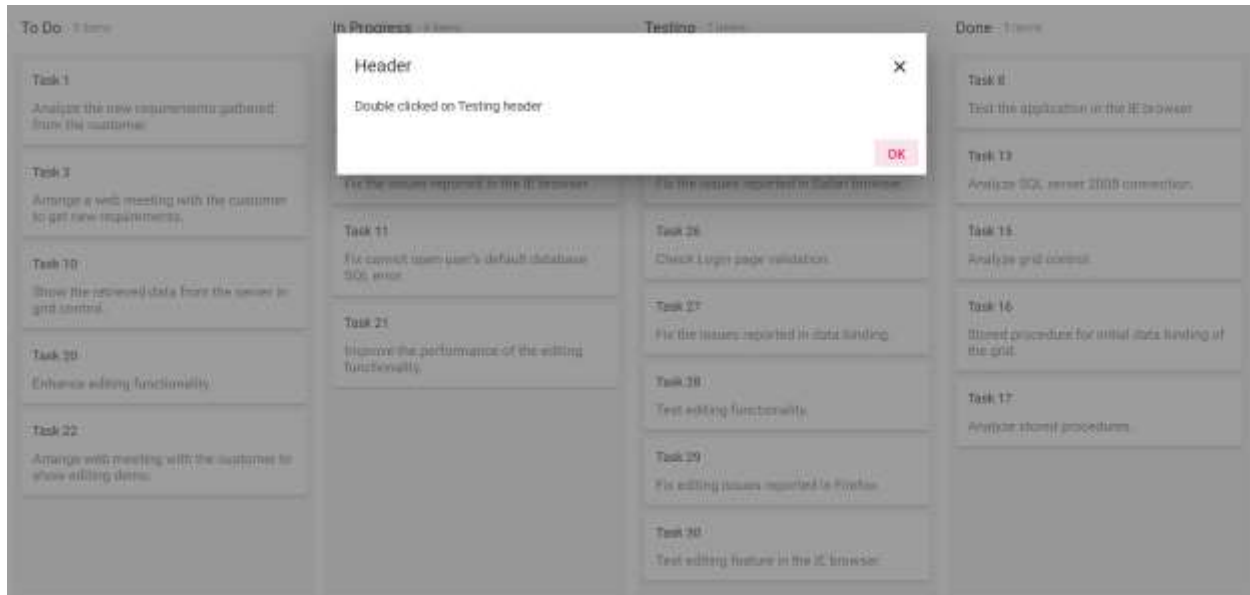
```

```

reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
    return TaskDetails;
}
}

```

Output be like the below.



Change Kanban columns dynamically

You can dynamically change the Kanban columns by using the [columns](#) property.

In the below sample, you can dynamically change the [allowToggle](#) property at the particular column when you click on the button. You can also change the initially created columns to the new Kanban columns by using the [columns](#) property when you click on the button.

CSHTML

```
@Html.EJS().Button("particularColumn").Content("Enable Allow Toggle").Render()
@Html.EJS().Button("column").Content("Change Columns").Render()
@Html.EJS().Kanban("kanban").KeyField("Status").Created("onCreate").DataSource(
    (IEnumerable<object>) ViewBag.data).Columns(col =>
    {
        col.HeaderText("To Do").KeyField("Open").Add();
        col.HeaderText("In Progress").KeyField("InProgress").Add();
        col.HeaderText("Testing").KeyField("Testing").Add();
        col.HeaderText("Done").KeyField("Close").Add();
    }).CardSettings(card => {
        card.ContentField("Summary").HeaderField("Id").ShowHeader(false);
    }).Render()
<script type="text/javascript">
    var kanbanObj;
    function onCreate() {
        kanbanObj = this;
    }
    document.getElementById('particularColumn').onclick = function () {
        kanbanObj.columns[1].allowToggle= true;
    };
    document.getElementById('column').onclick = function () {
        kanbanObj.columns = [
            { headerText: 'To Do', keyField: 'Open' },
            { headerText: 'Done', keyField: 'Close' }
        ]
    };
</script>
```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public Double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title = "Task
- 29001", Status = "Open", Summary = "Analyze the new requirements gathered
from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title = "Task
- 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title = "Task
- 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title = "Task
- 29004", Status = "InProgress", Summary = "Fix the issues reported in the
IE browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE",
Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000"
});
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title = "Task
- 29005", Status = "Review", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate =
3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title = "Task
- 29007", Status = "Validate", Summary = "Validate new requirements", Type =
"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title = "Task
- 29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix, Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title = "Task
- 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    }
}

```



```

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title = "Task
- 29011", Status = "Validate", Summary = "Validate the issues reported by
the customer.", Type = "Story", Priority = "High", Tags = "Validation,Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title = "Task
- 29015", Status = "Open", Summary = "Show the retrieved data from the
server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title = "Task
- 29016", Status = "InProgress", Summary = "Fix cannot open user's default
database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title = "Task
- 29017", Status = "Review", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title = "Task
- 29018", Status = "Close", Summary = "Analyze SQL server 2008 connection.",
Type = "Story", Priority = "Release Breaker", Tags = "Grid,Sql", Estimate =
2, Assignee = "Andrew Fuller", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title = "Task
- 29019", Status = "Validate", Summary = "Validate databinding issues.",
Type = "Story", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title = "Task
- 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title = "Task
- 29021", Status = "Close", Summary = "Stored procedure for initial data
binding of the grid.", Type = "Others", Priority = "Release Breaker", Tags =
"Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId = 6, Color
= "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title = "Task
- 29022", Status = "Close", Summary = "Analyze stored procedures.", Type =
"Story", Priority = "Release Breaker", Tags = "Procedures", Estimate = 5.5,
Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title = "Task
- 29023", Status = "Validate", Summary = "Validate editing issues.", Type =
"Story", Priority = "Critical", Tags = "Editing", Estimate = 1, Assignee =
"Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title = "Task
- 29024", Status = "Review", Summary = "Test editing functionality.", Type =
"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title = "Task
- 29025", Status = "Open", Summary = "Enhance editing functionality.", Type
= "Improvement", Priority = "Low", Tags = "Editing", Estimate = 3.5,
Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title = "Task
- 29026", Status = "InProgress", Summary = "Improve the performance of the
editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title = "Task
- 29027", Status = "Open", Summary = "Arrange web meeting with the customer
to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title = "Task
- 29029", Status = "Review", Summary = "Fix the editing issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Editing,Fix",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6, Color = "#cc0000"
});

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title = "Task
- 29030", Status = "Testing", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Critical", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title = "Task
- 29031", Status = "Testing", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title = "Task
- 29032", Status = "Testing", Summary = "Check Login page validation.", Type
= "Story", Priority = "Release Breaker", Tags = "Testing", Estimate = 0.5,
Assignee = "Michael Suyama", RankId = 3 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title = "Task
- 29033", Status = "Testing", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title = "Task
- 29034", Status = "Testing", Summary = "Test editing functionality.", Type
= "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5 });

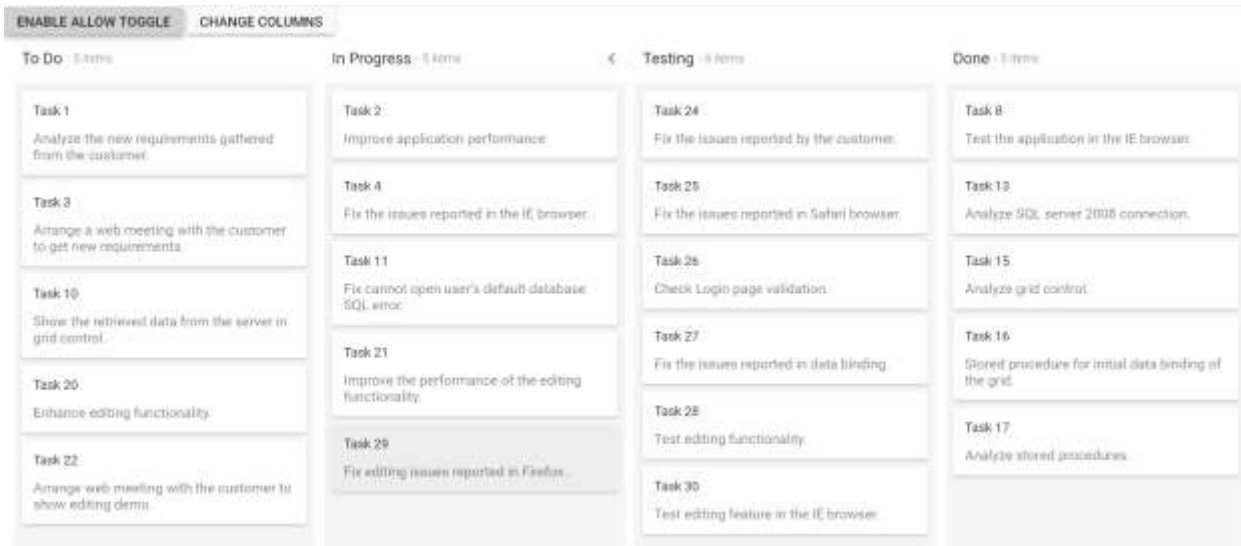
        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title = "Task
- 29035", Status = "Testing", Summary = "Fix editing issues reported in
Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title = "Task
- 29036", Status = "Testing", Summary = "Test editing feature in the IE
browser.", Type = "Story", Priority = "Normal", Tags = "Testing", Estimate =
5.5, Assignee = "Janet Leverling", RankId = 10 });

        return TaskDetails;
    }
}

```

Output be like the below.



Filtering Cards

You can filter the collection of cards from the dataSource and display it on the Kanban board by using the [query](#) property.

In the below sample, you can filter the cards based on the 'where' query and display the filtered data to the Kanban board.

CSHTML

```
@Html.EJS().DropDownList("priority_filter").Placeholder("Select a
priority").Index(0).Created("change").DataSource((IEnumerable<object>)ViewBag.
PriorityData).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "Priority"
}).Render()
@Html.EJS().Kanban("kanban").KeyField("Status").Created("onCreate").DataSou
rce((IEnumerable<object>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id").ShowHeader(false);
}).Render()
<script type="text/javascript">
    var kanbanObj;
    function onCreate() {
        kanbanObj = this;
    }
    function change(args) {
        var filterQuery = new ej.data.Query();
        if (args.value !== 'None') {
            filterQuery = new ej.data.Query().where('Priority', 'equal',
args.value);
        }
        kanbanObj.query = filterQuery;
    }
</script>
```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title = "Task
- 29001", Status = "Open", Summary = "Analyze the new requirements gathered
from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title = "Task
- 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title = "Task
- 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title = "Task
- 29004", Status = "InProgress", Summary = "Fix the issues reported in the
IE browser.", Type = "Bug", Priority = "Release Breaker", Tags = "IE",
Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color = "#cc0000"
});
        TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title = "Task
- 29005", Status = "Review", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate =
3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title = "Task
- 29007", Status = "Validate", Summary = "Validate new requirements", Type =
"Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title = "Task
- 29009", Status = "Review", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix, Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color = "#cc0000"
});
        TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title = "Task
- 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
    }
}

```

```

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title = "Task
- 29011", Status = "Validate", Summary = "Validate the issues reported by
the customer.", Type = "Story", Priority = "High", Tags = "Validation,Fix",
Estimate = 1, Assignee = "Steven walker", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title = "Task
- 29015", Status = "Open", Summary = "Show the retrieved data from the
server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title = "Task
- 29016", Status = "InProgress", Summary = "Fix cannot open user's default
database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title = "Task
- 29017", Status = "Review", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});
TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title = "Task
- 29018", Status = "Close", Summary = "Analyze SQL server 2008 connection.",
Type = "Story", Priority = "Release Breaker", Tags = "Grid,Sql", Estimate =
2, Assignee = "Andrew Fuller", RankId = 4, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title = "Task
- 29019", Status = "Validate", Summary = "Validate databinding issues.",
Type = "Story", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title = "Task
- 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title = "Task
- 29021", Status = "Close", Summary = "Stored procedure for initial data
binding of the grid.", Type = "Others", Priority = "Release Breaker", Tags =
"Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId = 6, Color
= "#27AE60" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title = "Task
- 29022", Status = "Close", Summary = "Analyze stored procedures.", Type =
"Story", Priority = "Release Breaker", Tags = "Procedures", Estimate = 5.5,
Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title = "Task
- 29023", Status = "Validate", Summary = "Validate editing issues.", Type =
"Story", Priority = "Critical", Tags = "Editing", Estimate = 1, Assignee =
"Nancy Davloio", RankId = 1, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title = "Task
- 29024", Status = "Review", Summary = "Test editing functionality.", Type =
"Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title = "Task
- 29025", Status = "Open", Summary = "Enhance editing functionality.", Type
= "Improvement", Priority = "Low", Tags = "Editing", Estimate = 3.5,
Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297" });
TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title = "Task
- 29026", Status = "InProgress", Summary = "Improve the performance of the
editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

```

```

        TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title = "Task
- 29027", Status = "Open", Summary = "Arrange web meeting with the customer
to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title = "Task
- 29029", Status = "Review", Summary = "Fix the editing issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Editing,Fix",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6, Color = "#cc0000"
});

        TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title = "Task
- 29030", Status = "Testing", Summary = "Fix the issues reported by the
customer.", Type = "Bug", Priority = "Critical", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title = "Task
- 29031", Status = "Testing", Summary = "Fix the issues reported in Safari
browser.", Type = "Bug", Priority = "Release Breaker", Tags = "Fix,Safari",
Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title = "Task
- 29032", Status = "Testing", Summary = "Check Login page validation.", Type
= "Story", Priority = "Release Breaker", Tags = "Testing", Estimate = 0.5,
Assignee = "Michael Suyama", RankId = 3 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title = "Task
- 29033", Status = "Testing", Summary = "Fix the issues reported in data
binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title = "Task
- 29034", Status = "Testing", Summary = "Test editing functionality.", Type
= "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5 });

        TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title = "Task
- 29035", Status = "Testing", Summary = "Fix editing issues reported in
Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

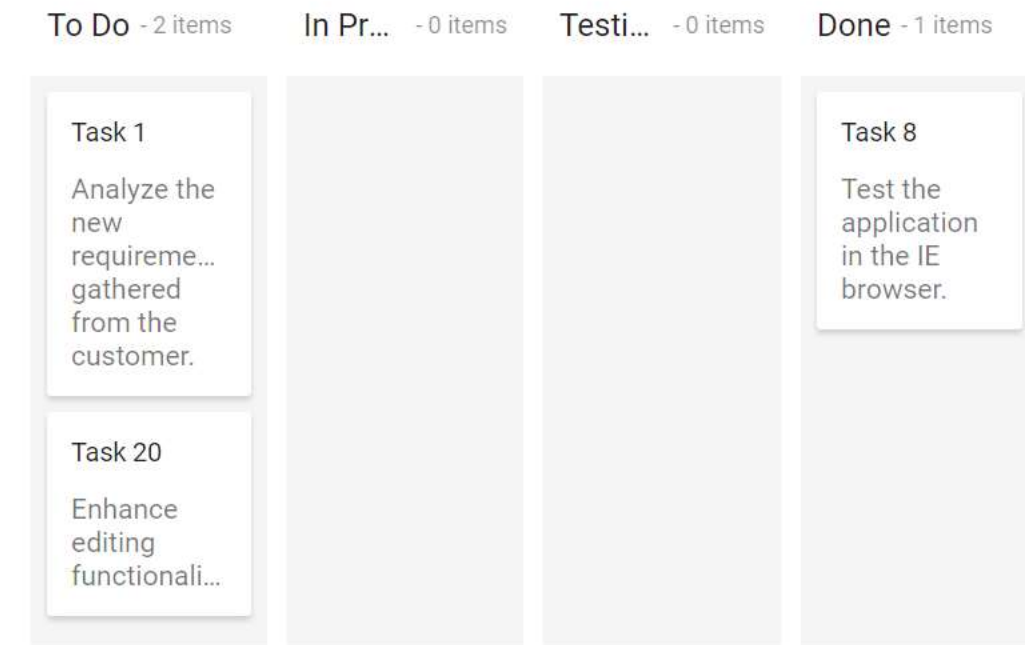
        TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title = "Task
- 29036", Status = "Testing", Summary = "Test editing feature in the IE
browser.", Type = "Story", Priority = "Normal", Tags = "Testing", Estimate =
5.5, Assignee = "Janet Leverling", RankId = 10 });

        return TaskDetails;
    }
}

```

Output be like the below.

Low



Searching Cards

You can search the cards in Kanban by using the `query` property.

In the following sample, the searching operation starts as soon as you start typing characters in the external text box. It will search the cards based on the `Id` and `Summary` using the `search` query with `contains` operator.

C#HTML

```
<table>
  <tbody>
    <td style="width: 200px">
      @Html.EJS().TextBox("search_text").Placeholder("Enter search
text").ShowClearButton(true).Created("searchCreate").Render();
    </td>
    <td>
      @Html.EJS().Button("reset_filter").Content("Reset").Render()
    </td>
  </tbody>
</table>

@Html.EJS().Kanban("kanban").KeyField("Status").DataSource((IEnumerable<obje
ct>)ViewBag.data).Columns(col =>
{
    col.HeaderText("To Do").KeyField("Open").Add();
    col.HeaderText("In Progress").KeyField("InProgress").Add();
    col.HeaderText("Testing").KeyField("Testing").Add();
    col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card => {
    card.ContentField("Summary").HeaderField("Id");
}).Created("kanbanCreate").Render()

<script>
```

```

var kanbanObj, textObj;
function kanbanCreate() {
    kanbanObj = this;
}
function searchCreate() {
    textObj = this;
}
function change(args) {
    var filterQuery = new ej.data.Query();
    if (args.value !== 'None') {
        filterQuery = new ej.data.Query().where('Priority', 'equal',
args.value);
    }
    kanbanObj.query = filterQuery;
}
document.getElementById('reset_filter').onclick = function () {
    textObj.value = '';
    kanbanObj.query = new ej.data.Query();
};
document.getElementById('search_text').onkeyup = function (e) {
    var searchValue = e.target.value;
    var searchQuery = new ej.data.Query();
    if (searchValue !== '') {
        searchQuery = new ej.data.Query().search(searchValue, ['Id',
'Summary'], 'contains', true);
    }
    kanbanObj.query = searchQuery;
};
</script>

```

DATASOURCE.CS

```

public class KanbanDataModels
{
    public string Id { get; set; }
    public string Title { get; set; }
    public string Status { get; set; }
    public string Summary { get; set; }
    public string Type { get; set; }
    public string Priority { get; set; }
    public string Tags { get; set; }
    public double Estimate { get; set; }
    public string Assignee { get; set; }
    public int RankId { get; set; }
    public string Color { get; set; }
    public List<KanbanDataModels> KanbanTasks()
    {
        List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
        TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title =
"Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
        TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title =
"Task - 29002", Status = "InProgress", Summary = "Improve application

```



```

performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title =
"Task - 29003", Status = "Open", Summary = "Arrange a web meeting with the
customer to get new requirements.", Type = "Others", Priority = "Critical",
Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling", RankId = 2,
Color = "#27AE60" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title =
"Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title =
"Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title =
"Task - 29007", Status = "Validate", Summary = "Validate new requirements",
Type = "Improvement", Priority = "Low", Tags = "Validation", Estimate = 1.5,
Assignee = "Robert King", RankId = 1, Color = "#7d7297" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title =
"Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title =
"Task - 29010", Status = "Close", Summary = "Test the application in the IE
browser.", Type = "Story", Priority = "Low", Tags = "Review,IE", Estimate =
5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title =
"Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title =
"Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title =
"Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title =
"Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title =
"Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title =
"Task - 29019", Status = "Validate", Summary = "Validate databinding

```

```

issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title =
"Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type =
"Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title =
"Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title =
"Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title =
"Task - 29023", Status = "Validate", Summary = "Validate editing issues.",
Type = "Story", Priority = "Critical", Tags = "Editing", Estimate = 1,
Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title =
"Task - 29024", Status = "Review", Summary = "Test editing functionality.",
Type = "Story", Priority = "Normal", Tags = "Editing,Test", Estimate = 0.5,
Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title =
"Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});
    TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title =
"Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title =
"Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title =
"Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title =
"Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title =
"Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title =
"Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });
    TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title =
"Task - 29033", Status = "Testing", Summary = "Fix the issues reported in

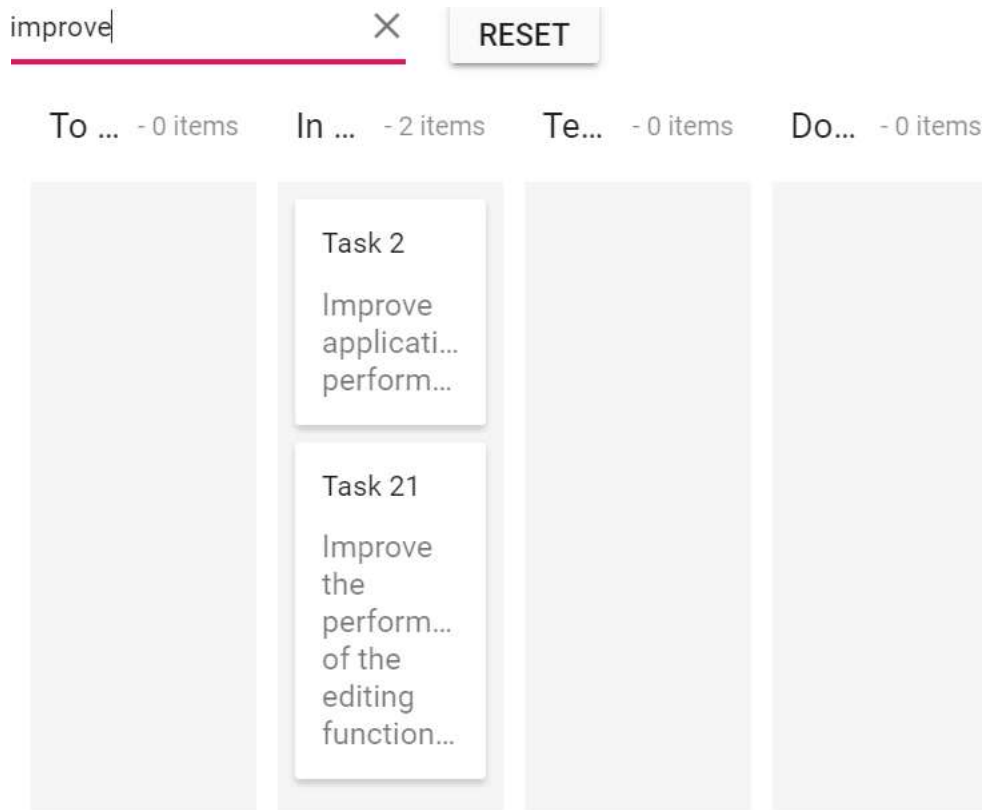
```

```

data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title =
"Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title =
"Task - 29035", Status = "Testing", Summary = "Fix editing issues reported
in Firefox.", Type = "Bug", Priority = "Critical", Tags = "Editing,Fix",
Estimate = 1.5, Assignee = "Robert King", RankId = 7 });
TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title =
"Task - 29036", Status = "Testing", Summary = "Test editing feature in the
IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });
return TaskDetails;
}
}

```

Output be like the below.



Linear Gauge

Getting Started with ASP.NET MVC Linear Gauge Control

This section briefly explains about how to include [ASP.NET MVC Linear Gauge](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

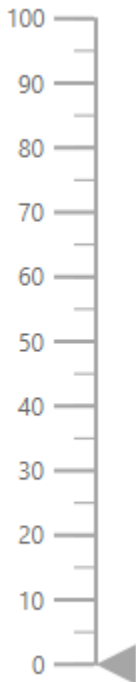
Add ASP.NET MVC Linear Gauge control

Now, add the Syncfusion ASP.NET MVC Linear Gauge control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().LinearGauge("container").Render();
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Linear Gauge control will be rendered in the default web browser.

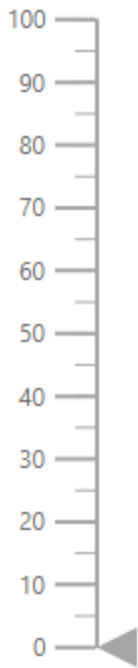
**Add Gauge Title**

The title for the Linear Gauge can be set using [Title](#) property in Linear Gauge.

CSHTML

```
@Html.EJS().LinearGauge("container").Title("Temperature Measure").Render();
```

Temperature Measure

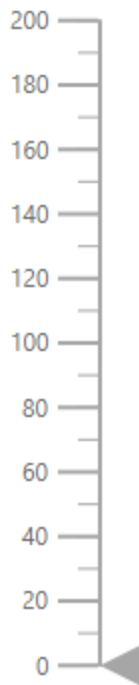


Axis

The start value and end value for the Linear Gauge can be set using the [Minimum](#) and [Maximum](#) properties in the `axis` object.

C#HTML

```
@Html.EJS().LinearGauge("container").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Minimum = 0, Maximum = 200
    })}.Render();
```



Note: [View Sample in GitHub.](#)

Note: You can also explore our [ASP.NET MVC Linear Gauge example](#) that shows you how to render the Linear Gauge in ASP.NET MVC.

Dimensions in ASP.NET MVC Linear Gauge

<!-- markdownlint-disable MD036 -->

Size for Linear Gauge

The height and width of the Linear Gauge can be set using the [Height](#) and [Width](#) properties in [LinearGauge](#).

In Pixel

The size of the Linear Gauge can be set in pixel as demonstrated below.

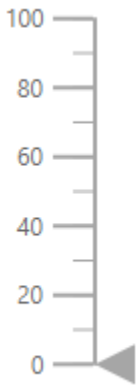
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;  
@Html.EJS().LinearGauge("linear").Width("650px").Height("350px").Render()
```

PIXEL.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
using Syncfusion.EJ2.Charts;
```

```
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



In Percentage

By setting value in percentage, Linear Gauge receives its dimension matching to its parent. For example, when the height is set as **50%**, Linear Gauge renders to half of the parent height. The Linear Gauge will be responsive when the width is set as **100%**.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").Width("100%").Height("50%").Render()
```

PERCENTAGE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {

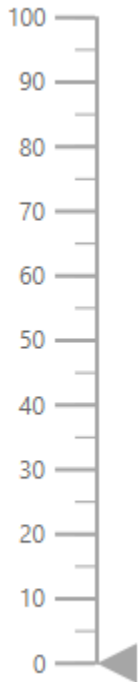
```



```

        return View();
    }
}

```



Note: When the component's size is not specified, the height will be **450px** and the width will be the same as the parent element's width.

Axis in ASP.NET MVC Linear Gauge

Axis is used to indicate the numeric values in the linear scale. The Linear Gauge component can have any number of axes. The sub-elements of an axis are line, ticks, labels, ranges, and pointers.

Setting the start value and end value of the axis

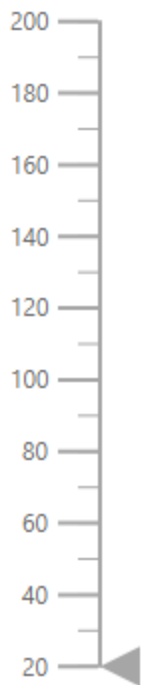
The start value and end value for the Linear Gauge can be set using the [Minimum](#) and [Maximum](#) properties in the [LinearGaugeAxis](#) respectively. By default, the start value of the axis is **0** and the end value of the axis is **100**.

CSHTML

```

@using Syncfusion.EJ2.LinearGauge
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Minimum=20, Maximum=200
    }
}).Render()

```



Line Customization

The following properties in the [LinearGaugeLine](#) can be used to customize the axis line in the Linear Gauge.

- [Height](#) - To set the length of the axis line.
- [Width](#) - To set the thickness of the axis line.
- [Color](#) - To set the color of the axis line.
- [Offset](#) - To render the axis line with the specified distance from the Linear Gauge.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;  
@Html.EJS().LinearGauge("gauge").Axes(new  
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>  
{  
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis  
    {  
        Line = new Syncfusion.EJ2.LinearGauge.LinearGaugeLine  
        {  
            Height = 150, Width= 2, Color = "#4286f4", Offset = 2  
        }  
    }  
}).Render()
```

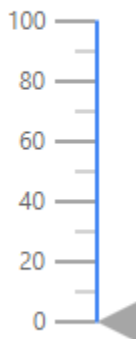
LINE-CUSTOMIZATION.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Ticks Customization

Ticks are used to specify the interval in the axis. Ticks are of two types, major ticks and minor ticks. The following properties in the [LinearGaugeMajorTicks](#) and [LinearGaugeMinorTicks](#) can be used to customize the major ticks and minor ticks respectively.

- [Height](#) - To set the length of the major and minor ticks in pixel values.
- [Color](#) - To set the color of the major and minor ticks of the Linear Gauge.
- [Width](#) - To set the thickness of the major and minor ticks in pixel values.
- [Interval](#) - To set the interval for the major ticks and minor ticks in the Linear Gauge.

<!-- markdownlint-disable MD036 -->

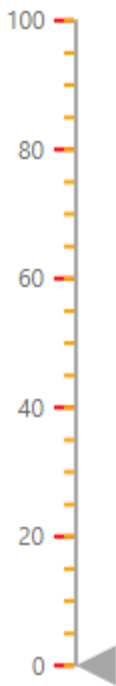
CSHTML

```
@using Syncfusion.EJ2;
```

```
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        MajorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
        {
            Height=10, Width=2, Interval=20, Color="red"
        },
        MinorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
        {
            Height = 5, Width = 2, Interval = 5, Color = "Orange"
        }
    }
}).Render()
```

TICKS-CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Positioning the ticks

The minor and major ticks can be positioned by using the [Offset](#) and [Position](#) properties. The [Offset](#) is used to render the ticks with the specified distance from the axis. By default, the offset value is **0**. The possible values of the [Position](#) property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, the ticks will be placed inside the axis.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        MajorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
        {
            Height=10, Width=2, Interval=20, Color="red", Position =
Position.Outside
        },
        MinorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
        {
            Height = 5, Width = 2, Interval = 5, Color = "Orange", Position
= Position.Cross
        }
    }
}).Render()
```

TICK-POSITION.CS

```
using Microsoft.AspNetCore.Mvc;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Labels Customization

The style of the labels can be customized using the following properties in [LinearGaugeFont](#) in the [LinearGaugeLabel](#).

- [Color](#) - To set the color of the axis label.
- [FontFamily](#) - To set the font family of the axis label.
- [FontStyle](#) - To set the font style of the axis label.
- [FontWeight](#) - To set the font weight of the axis label.
- [Opacity](#) - To set the opacity of the axis label.
- [Size](#) - To set the size of the axis label.

<!-- markdownlint-disable MD036 -->

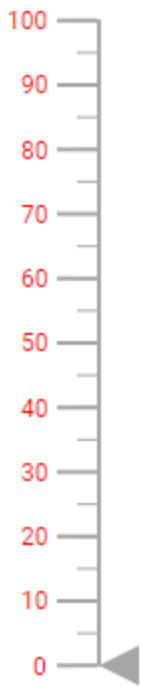
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
```

```
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
        {
            Font = new LinearGaugeFont
            {
                FontFamily = "Roboto",
                Size = "12px",
                Color = "red",
                Opacity = 1,
                FontWeight = "Regular"
            }
        }
    }
}).Render()
```

LABELS-CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Positioning the axis label

Labels can be positioned by using [Offset](#) and [Position](#) properties in the [LinearGaugeLabel](#). The [Offset](#) defines the distance between the labels and ticks. By default, the offset value is **0**. The possible values of [Position](#) property are **Inside**, **Outside**, **Cross**, and **Auto**. By default, the labels will be placed inside the axis.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
        {
            Position = Position.Cross
        }
    }
}).Render()
```

LABEL-POSITION.CS

```
using Microsoft.AspNetCore.Mvc;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
        }
    }
}
```



```

    {
        return View();
    }
}

```



Customizing the display of the last label

If the last label is not in the visible range, it will be hidden by default. The last label can be made visible by setting the [ShowLastLabel](#) property as **true** in the [LinearGaugeAxis](#).

<!-- markdownlint-disable MD036 -->

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Minimum = 0, Maximum = 115, ShowLastLabel = true
    }
}).Render()

```

SHOWLASTLABEL.CS

```

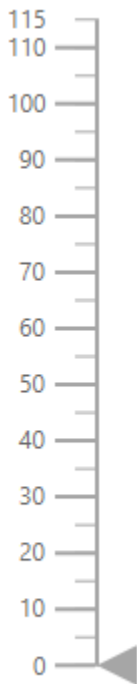
using System;
using System.Collections.Generic;
using System.Diagnostics;

```

```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



<!-- markdownlint-disable MD036 -->

Label Format

Axis labels in the Linear Gauge control can be formatted using the [Format](#) property in [LinearGaugeLabel](#). It is used to render the axis labels in a certain format or to add a user-defined unit in the label. It works with the help of placeholder like **{value}°C**, where **value** represents the axis value. For example, 20°C.

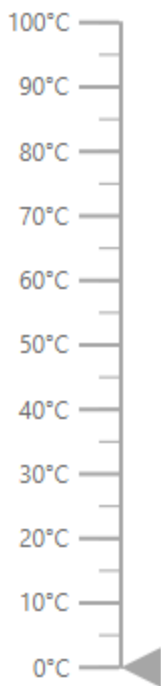
CSHTML

```
@using Syncfusion.EJ2;
```

```
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
        {
            Format = "{value}°C"
        }
    }
}).Render()
```

LABEL-FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Displaying numeric format in labels

The numeric formats such as currency, percentage, and so on can be displayed in the labels of the Linear Gauge using the [Format](#) property in the Linear Gauge component. The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

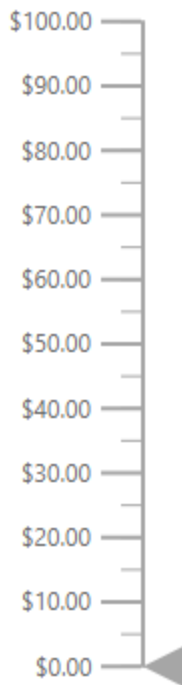
| Label Value | Label Format property value | Result | Description |
|-------------|-----------------------------|------------|---|
| 1000 | n1 | 1000.0 | The number is rounded to 1 decimal place. |
| 1000 | n2 | 1000.00 | The number is rounded to 2 decimal place. |
| 1000 | n3 | 1000.000 | The number is rounded to 3 decimal place. |
| 0.01 | p1 | 1.0% | The number is converted to percentage with 1 decimal place. |
| 0.01 | p2 | 1.00% | The number is converted to percentage with 2 decimal place. |
| 0.01 | p3 | 1.000% | The number is converted to percentage with 3 decimal place. |
| 1000 | c1 | \$1,000.0 | The currency symbol is appended to number and number is rounded to 1 decimal place. |
| 1000 | c2 | \$1,000.00 | The currency symbol is appended to number and number is rounded to 2 decimal place. |

CHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Format("c").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
    }
}) .Render()
```

CUSTOM-LABEL.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Orientation

By default, the Linear Gauge is rendered vertically. To change its orientation, the [Orientation](#) property must be set to **Horizontal**.

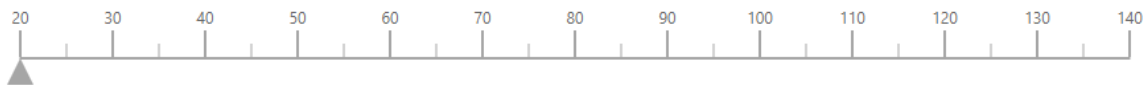
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Orientation(Orientation.Horizontal).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Minimum = 20, Maximum = 140
    }
}).Render()
```

ORIENTATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
```

```
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Inverted Axis

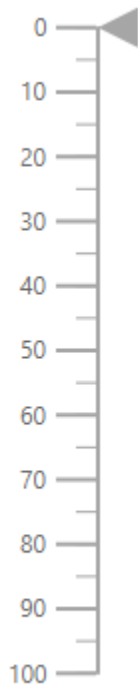
The axis of the Linear Gauge component can be inverted by setting the [IsInversed](#) property to **true** in the [LinearGaugeAxis](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        IsInversed = true
    }
}).Render()
```

INVERTED-AXES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Opposed Axis

To place an axis opposite from its original position, [OpposedPosition](#) property in the [LinearGaugeAxis](#) must be set as **true**.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        OpposedPosition = true
    }
}).Render()
```

OPPOSED-AXES.CS

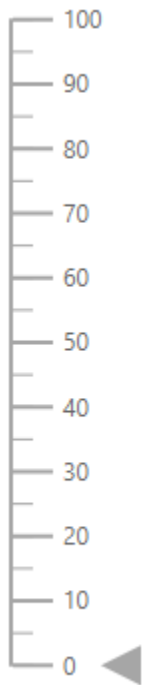
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
```



```

namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Multiple Axes

Multiple axes can be added to the Linear Gauge by adding multiple [LinearGaugeAxis](#) in the [LinearGaugeAxes](#) and customization can be done with the [LinearGaugeAxis](#). Each axis can be customized separately as shown in the following example.

CSHTML

```

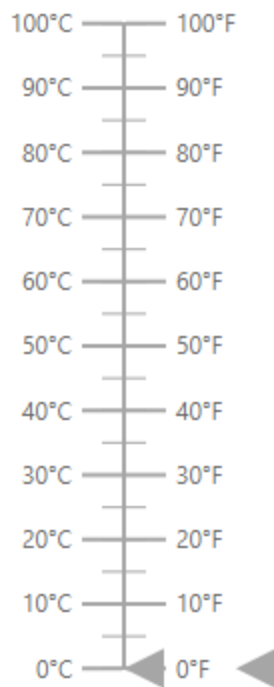
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
        {
            Format="{value}°C"
        }
    },
}

```

```
new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
{
    OpposedPosition = true,
    LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
    {
        Format="{value}°F"
    }
}
}).Render()
```

MULTIPLE-AXES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Ranges in ASP.NET MVC Linear Gauge

Range is the set of values in the axis. The range can be defined using the [Start](#) and [End](#) properties in the [LinearGaugeRange](#). Any number of ranges can be added to the Linear Gauge using the [LinearGaugeRanges](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Ranges = new List<LinearGaugeRange>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugeRange
            {
                Start = 50, End = 80
            }
        }
    }
}).Render();
```

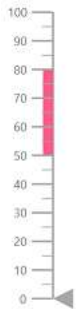
RANGES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```

```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Customizing the range

Ranges can be customized using the following properties in [LinearGaugeRange](#).

- [StartWidth](#) - Customize the range thickness at the start axis value.
- [EndWidth](#) - Customize the range thickness at the end axis value.
- [Color](#) - Customize the range color.
- [Position](#) - To place the range. By default, the range is placed outside of the axis. To change the position, this property can be set as "Inside", "Outside", "Cross", or "Auto".
- [Offset](#) - To place the range with specified distance from the axis.
- [Border](#) - Customize color and width of range border.

CSHTML

```

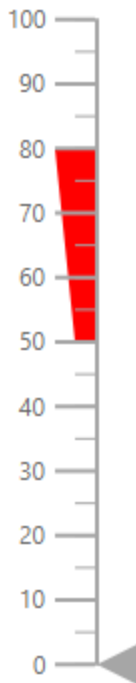
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Ranges = new List<LinearGaugeRange>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugeRange
            {

```

```
                Start = 20, End = 60, StartWidth = 10, EndWidth = 20, Color  
= "red"  
            }  
        }  
    }  
}).Render();
```

RANGE-CUSTOMIZATION.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
using Syncfusion.EJ2.Charts;  
using Syncfusion.EJ2.LinearGauge;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            return View();  
        }  
    }  
}
```



Setting the range color for the labels

To set the color of the labels like the range color, set the [UseRangeColor](#) property as **true** in the [LinearGaugeLabel](#).

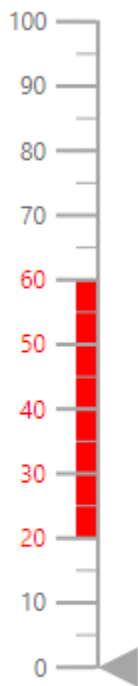
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
        {
            UseRangeColor = true
        },
        Ranges = new List<LinearGaugeRange>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugeRange
            {
                Start = 20, End = 60, Color = "red"
            }
        }
    }
}).Render();
```

RANGE-LABELCOLOR.CS

```
using Microsoft.AspNetCore.Mvc;
```

```
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Multiple ranges

Multiple ranges can be added to the Linear Gauge by adding collections of [LinearGaugeRange](#) in the [LinearGaugeRanges](#) and customization of ranges can be done with [LinearGaugeRange](#).

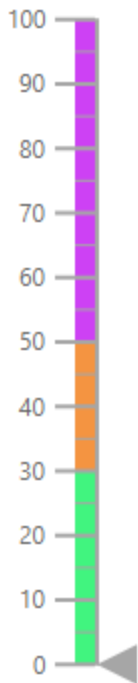
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Ranges = new List<LinearGaugeRange>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugeRange
            {
```

```
                Start = 0, End = 30, Color = "#41f47f", StartWidth = 10,
EndWidth = 10
            },
            new Syncfusion.EJ2.LinearGauge.LinearGaugeRange
            {
                Start = 30, End = 50, Color = "#f49441", StartWidth = 10,
EndWidth = 10
            },
            new Syncfusion.EJ2.LinearGauge.LinearGaugeRange
            {
                Start = 50, End = 100, Color = "#cd41f4", StartWidth = 10,
EndWidth = 10
            }
        }
    }
}).Render();
```

MULTIPLE-RANGES.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Gradient Color

Gradient support allows the addition of multiple colors in the range. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear-gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [StartValue](#) property. The end value of the linear gradient will be set using the [EndValue](#) property. The color stop values such as [Color](#), [Opacity](#), and [Offset](#) to be defined in [ColorStop](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Load("onGaugeLoad").Orientation(Orientation.Horizontal).Container(
    new Syncfusion.EJ2.LinearGauge.LinearGaugeContainer {
        Width = 30, Offset = 30 }
).Margin(new Syncfusion.EJ2.LinearGauge.LinearGaugeMargin{
    Bottom=50}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Minimum = 0, Maximum = 100,
        Line = new Syncfusion.EJ2.LinearGauge.LinearGaugeLine {
            Width = 0 },
```

```

        MajorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
    { Interval= 25, Height= 0 },
        MinorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
    { Height=0 },
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
    { Offset = 55 },
    }
    }.Render();
<script type="text/javascript">
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].pointers = [{
            value: 80, height: 25,
            width: 35, placement: 'Near',
            offset: -44, markerType: 'Triangle',
            color: '#f54ea2'
        }];
        sender.gauge.axes[0].ranges = [{
            start: 0, end: 80,
            startWidth: 30, endWidth: 30,
            offset: 30,
            linearGradient: {
                startValue: '0%',
                endValue: '100%',
                colorStop: [
                    { color: '#fef3f9', offset: '0%', opacity: 1 },
                    { color: '#f54ea2', offset: '100%', opacity: 1 }]
            }
        }]
    }
</script>

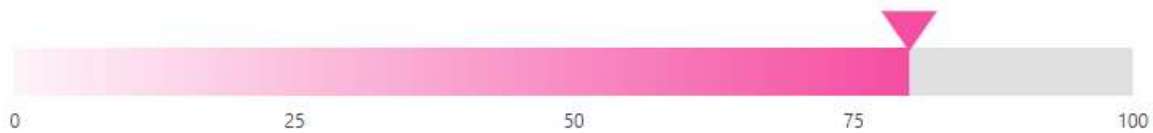
```

LINEARGRADIENT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [InnerPosition](#) property. The outer circle position of the radial gradient can be set using the [OuterPosition](#) property. The color stop values such as [Color](#), [Opacity](#), and [Offset](#) to be defined in [ColorStop](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Load("onGaugeLoad").Orientation(Orienta
tion.Horizontal).Container(
    new Syncfusion.EJ2.LinearGauge.LinearGaugeContainer
    {
        Width = 30, Offset = 30
    }).Margin(new Syncfusion.EJ2.LinearGauge.LinearGaugeMargin { Bottom = 50
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
    {
        new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
        {
            Minimum = 0, Maximum = 100,
            Line = new Syncfusion.EJ2.LinearGauge.LinearGaugeLine {
Width = 0 },
            MajorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
{ Interval= 25, Height= 0 },
            MinorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
{ Height=0 },
            LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
{ Offset = 55 },
        }
    }
).Render();
<script type="text/javascript">
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].pointers = [{
            value: 80, height: 25,
            width: 35, placement: 'Near',
            offset: -44, markerType: 'Triangle',
            color: '#f54ea2'
        }];
        sender.gauge.axes[0].ranges = [{
            start: 0, end: 80,
            startWidth: 30, endWidth: 30,
            offset: 30,
            radialGradient: {
                radius: '60%',
                outerPosition: { x: '50%', y: '50%' },
                innerPosition: { x: '50%', y: '50%' },
            }
        }];
    }
}
```

```

        colorStop: [
            { color: '#fff5f5', offset: '0%', opacity: 0.9 },
            { color: '#f54ea2', offset: '100%', opacity: 0.8 }]
        }
    }
}
</script>

```

RADIALGRADIENT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Note: If we set both gradients for the range, only the linear gradient gets rendered. If we set the [StartValue](#) and [EndValue](#) of the [LinearGradient](#) as empty strings, then the radial gradient gets rendered in the range of the Linear Gauge.

Pointers in ASP.NET MVC Linear Gauge

Pointers are used to indicate values on an axis. The value of the pointer can be modified using the [Value](#) property in [LinearGaugePointer](#).

CSHTML

```

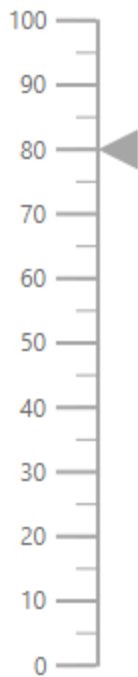
@using Syncfusion.EJ2;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{

```

```
new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
{
    Pointers = new List<LinearGaugePointer>
    {
        new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
        {
            Value = 80
        }
    }
}).Render()
```

POINTERS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Types of pointer

The Linear Gauge supports the following types of pointers:

- Bar
- Marker

The type of pointer can be modified by using the [Type](#) property in [LinearGaugePointer](#).

Marker pointer

A marker pointer is a shape that can be used to mark the pointer value in the Linear Gauge.

Types of marker shapes

By default, the marker shape for the pointer is **InvertedTriangle**. To change the shape of the pointer, use the [MarkerType](#) property in [LinearGaugePointer](#). The following marker types are available in Linear Gauge.

- Circle
- Rectangle
- Triangle
- InvertedTriangle
- Diamond
- Image
- Text

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
```

```
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 60, Type = Point.Marker, MarkerType =
MarkerType.Circle
            }
        }
    }
}).Render()
```

MARKER-POINTER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

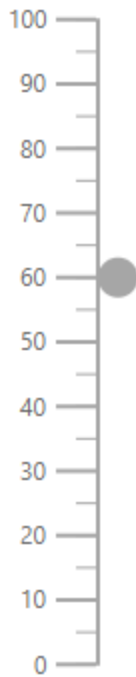


Image can be rendered instead of rendering a shape as a pointer. It can be achieved by setting the [MarkerType](#) property to **Image** and setting the source URL of image to [ImageUrl](#) property in [Pointer](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Orientation(Syncfusion.EJ2.LinearGauge.Orie
ntation.Horizontal).Axes(axes => axes.Minimum(0).Maximum(100)
    .LabelStyle(labelStyle => labelStyle.Font(font =>
font.FontFamily("inherit")).Position(Syncfusion.EJ2.LinearGauge.Position.Out
side))
    .Pointers(pointer =>
    {

pointer.Value(60).MarkerType(Syncfusion.EJ2.LinearGauge.MarkerType.Image).Wi
dth(40).Height(40).ImageUrl("https://ej2.syncfusion.com/aspnetmvc/Content/Li
nearGauge/step-count.png").Offset("-27").Add();
    })
    .MajorTicks(majorTick =>
majorTick.Interval(20).Height(7).Width(1).Position(Syncfusion.EJ2.LinearGaug
e.Position.Outside)).MinorTicks(minorTick =>
minorTick.Height(3).Width(1).Position(Syncfusion.EJ2.LinearGauge.Position.Ou
tside)).Add()).Render()
```

MARKER-POINTER-IMAGE.CS

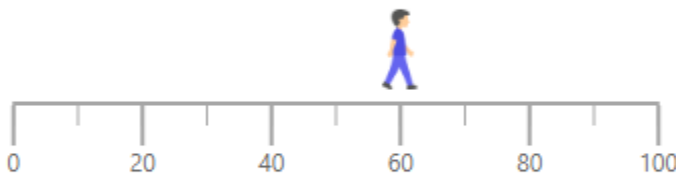
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
```



```

using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Text can be added instead of rendering a shape as a pointer. It can be achieved by setting the [MarkerType](#) property to **Text**, and the text content can be set using the [Text](#) property in [Pointer](#).

The following properties in the [TextStyle](#) property can be used to set the text style for the text pointer.

- [FontFamily](#) - It is used to set the font family for the text pointer.
- [FontStyle](#) - It is used to set the font style for the text pointer.
- [FontWeight](#) - It is used to set the font weight for the text pointer.
- [Size](#) - It is used to set the font size for the text pointer.

CSHTML

```

@using Syncfusion.EJ2.LinearGauge;
@{
    var textStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeFont { Size =
    "18px", FontWeight = "bold" };
}
@Html.EJS().LinearGauge("gauge").Orientation(Syncfusion.EJ2.LinearGauge.Orie
ntation.Horizontal).Axes(axes => axes.Minimum(0).Maximum(100)
    .Line(line => line.Width(0)).LabelStyle(labelStyle =>
labelStyle.Font(font =>

```

```

font.FontFamily("inherit")).Position(Syncfusion.EJ2.LinearGauge.Position.Outside))
    .Pointers(pointer =>
    {

pointer.Value(13).MarkerType(Syncfusion.EJ2.LinearGauge.MarkerType.Text).Text("Low").Color("Black").Offset("-35").TextStyle(textStyle).Add();

pointer.Value(48).MarkerType(Syncfusion.EJ2.LinearGauge.MarkerType.Text).Text("Moderate").Color("Black").Offset("-35").TextStyle(textStyle).Add();

pointer.Value(83).MarkerType(Syncfusion.EJ2.LinearGauge.MarkerType.Text).Text("High").Color("Black").Offset("-35").TextStyle(textStyle).Add();
    })
    .Ranges(range =>
    {

range.Start(0).End(30).StartWidth(50).EndWidth(50).Color("#FB7D55").Position(Syncfusion.EJ2.LinearGauge.Position.Outside).Add();

range.Start(30).End(65).StartWidth(50).EndWidth(50).Color("#ECC85B").Position(Syncfusion.EJ2.LinearGauge.Position.Outside).Add();

range.Start(65).End(100).StartWidth(50).EndWidth(50).Color("#6FC78A").Position(Syncfusion.EJ2.LinearGauge.Position.Outside).Add();
    }).MajorTicks(majorTick =>
majorTick.Interval(20).Height(7).Width(1).Position(Syncfusion.EJ2.LinearGauge.Position.Outside)).MinorTicks(minorTick =>
minorTick.Height(3).Interval(10).Position(Syncfusion.EJ2.LinearGauge.Position.Outside)).Add()).Render()

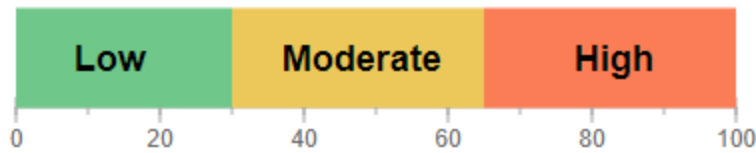
```

MARKER-POINTER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



<!-- markdownlint-disable MD036 -->

Marker pointer customization

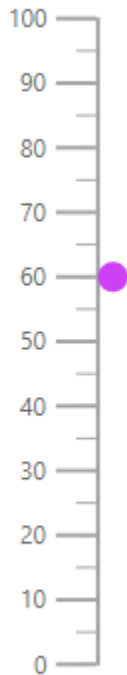
The marker pointer can be customized using the following properties and class.

- [Height](#) - To set the height of the pointer.
- [Position](#) - The position of the pointer can be changed by setting the value as **Inside**, **Outside**, **Cross**, or **Auto**.
- [Width](#) - To set the width of the pointer.
- [Color](#) - To set the color of the pointer.
- [Placement](#) - To place the pointer in the specified position. By default, the pointer is placed **Far** from the axis. To change the placement, set the [Placement](#) property as **Near**, **Center**, or **None**.
- [Offset](#) - To place the pointer with specified distance from the axis.
- [Opacity](#) - To set the opacity of the pointer.
- [AnimationDuration](#) - To specify the duration of the animation in pointer.
- [Border](#) - To set the color and width for the border of the pointer.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 60, Type = Point.Marker, MarkerType =
MarkerType.Circle, Height = 15, Width = 15, Color = "#cd41f4"
            }
        }
    }
})
```

```
}
}).Render()
```



Bar pointer

The bar pointer is used to track the axis value. The bar pointer starts from the beginning of the gauge and ends at the pointer value. To enable bar pointer set the [Type](#) property in [LinearGaugePointer](#) as **Bar**.

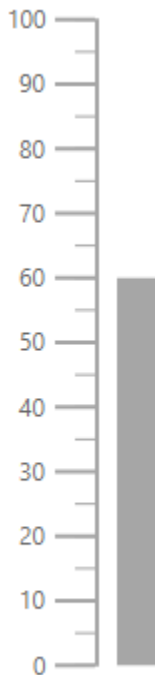
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 60, Type = Point.Bar
            }
        }
    }
}).Render()
```

BAR-POINTER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



<!-- markdownlint-disable MD036 -->

Bar pointer customization

The bar pointer can be customized using following properties and class.

- [Width](#) - To set the thickness of the bar pointer.
- [Color](#) - To set the color of the bar pointer.
- [Offset](#) - To place the bar pointer with the specified distance from it's default position.
- [Opacity](#) - To set the opacity of the bar pointer.

- [RoundedCornerRadius](#) - To set the corner radius of the bar pointer.
- [Border](#) - To set the color and width for the border of the pointer.
- [AnimationDuration](#) - To set the duration of the animation in bar pointer.

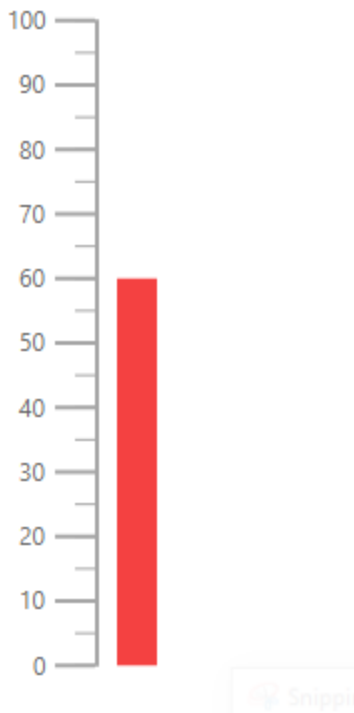
Note: The Placement property is not applicable for the bar pointer.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 60, Type = Point.Bar, Color = "#f44141"
            }
        }
    }
}).Render()
```

BAR-POINTER-CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Multiple pointers

Multiple pointers can be added to the Linear Gauge by adding multiple [LinearGaugePointer](#) in the [LinearGaugePointers](#) and customization for the pointers can be done with [LinearGaugePointer](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            },
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 30, MarkerType = MarkerType.Circle
            },
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 60, MarkerType = MarkerType.Diamond
            }
        }
    }
}).Render()
```

MULTIPLE-POINTERS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```

**Pointer animation**

Pointer is animated on loading the gauge. This can be handled using the [AnimationDuration](#) property. The duration of the animation can be specified in milliseconds.

CSHTML

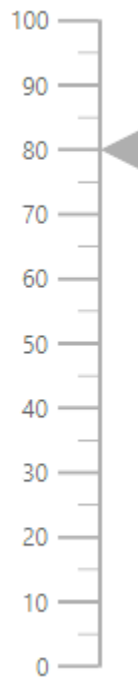
```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 60, AnimationDuration = 1000
            }
        }
    }
}).Render()

```


POINTER-ANIMATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

**Gradient Color**

Gradient support allows the addition of multiple colors in the pointers of the Linear Gauge. The following gradient types are supported in the Linear Gauge.

- Linear Gradient
- Radial Gradient

Linear Gradient

Using linear gradient, colors will be applied in a linear progression. The start value of the linear gradient can be set using the [StartValue](#) property. The end value of the linear gradient will be set using the [EndValue](#) property. The color stop values such as [Color](#), [Opacity](#), and [Offset](#) are set using [ColorStop](#) property.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;

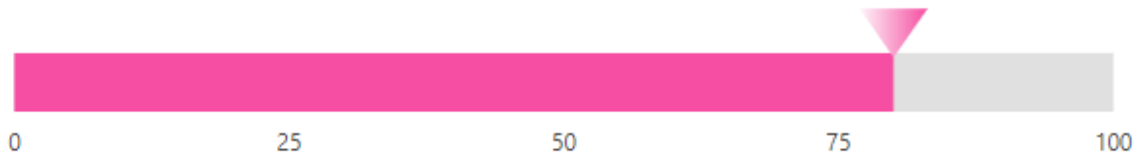
@Html.EJS().LinearGauge("container").Load("onGaugeLoad").Orientation(Orientation.Horizontal).Container(
    new Syncfusion.EJ2.LinearGauge.LinearGaugeContainer {
        Width = 30, Offset = 30
    }).Margin(new Syncfusion.EJ2.LinearGauge.LinearGaugeMargin {
        Bottom=50
    }).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
    {
        new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
        {
            Minimum = 0, Maximum = 100,
            Line = new Syncfusion.EJ2.LinearGauge.LinearGaugeLine {
                Width = 0 },
            MajorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
            { Interval= 25, Height= 0 },
            MinorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
            { Height=0 },
            LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
            { Offset = 55 },
        }
    }).Render();
<script type="text/javascript">
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].pointers = [{
            value: 80, height: 25,
            width: 35, placement: 'Near',
            offset: -44, markerType: 'Triangle',
            linearGradient: {
                startValue: '0%',
                endValue: '100%',
                colorStop: [
                    { color: '#fef3f9', offset: '0%', opacity: 1 },
                    { color: '#f54ea2', offset: '100%', opacity: 1 }]
            }
        }];
        sender.gauge.axes[0].ranges = [{
            start: 0, end: 80,
            startWidth: 30, endWidth: 30,
            color: '#f54ea2', offset: 30,
        }]
    }
</script>
```

LINEARGRADIENT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Radial Gradient

Using radial gradient, colors will be applied in circular progression. The inner circle position of the radial gradient will be set using the [InnerPosition](#) property. The outer circle position of the radial gradient can be set using the [OuterPosition](#) property. The color stop values such as [Color](#), [Opacity](#), and [Offset](#) are set using [ColorStop](#) property.

CSHTML

```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Load("onGaugeLoad").Orientation(Orientation.Horizontal).Container(
    new Syncfusion.EJ2.LinearGauge.LinearGaugeContainer {
        Width = 30, Offset = 30
    }).Margin(new Syncfusion.EJ2.LinearGauge.LinearGaugeMargin {
        Bottom=50
    }).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
    {
        new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
        {
            Minimum = 0, Maximum = 100,
            Line = new Syncfusion.EJ2.LinearGauge.LinearGaugeLine {
                Width = 0 },
            MajorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
            { Interval= 25, Height= 0 },

```

```

        MinorTicks = new Syncfusion.EJ2.LinearGauge.LinearGaugeTick
    { Height=0 },
        LabelStyle = new Syncfusion.EJ2.LinearGauge.LinearGaugeLabel
    { Offset = 55 },
    }
    }.Render();
<script type="text/javascript">
    function onGaugeLoad(sender) {
        sender.gauge.axes[0].pointers = [{
            value: 80, height: 25,
            width: 35, placement: 'Near',
            offset: -44, markerType: 'Triangle',
            radialGradient: {
                radius: '60%',
                outerPosition: { x: '50%', y: '50%' },
                innerPosition: { x: '50%', y: '50%' },
                colorStop: [
                    { color: '#fff5f5', offset: '0%', opacity: 0.9 },
                    { color: '#f54ea2', offset: '100%', opacity: 0.8 }]
            }
        }];
        sender.gauge.axes[0].ranges = [{
            start: 0, end: 80,
            startWidth: 30, endWidth: 30,
            color: '#f54ea2', offset: 30,
        }]
    }
</script>

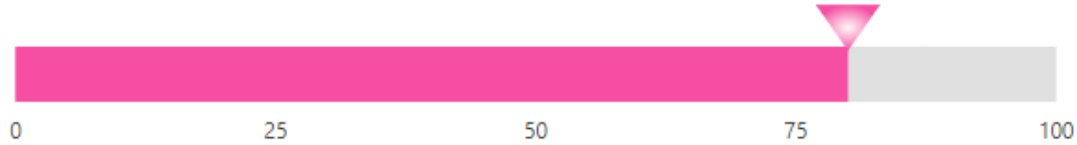
```

RADIALGRADIENT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Note: If we set both gradients, only the linear gradient gets rendered. If we set the [StartValue](#) and [EndValue](#) of the [LinearGradient](#) as empty strings, then the radial gradient gets rendered in the pointer of the Linear Gauge.

Annotations in ASP.NET MVC Linear Gauge

<!-- markdownlint-disable MD013 -->

Annotations are used to mark the specific area of interest in the Linear Gauge with text, HTML elements, or images. Any number of annotations can be added to the Linear Gauge component.

Adding annotation

To render the custom HTML elements in the Linear Gauge component, use the [Content](#) property in the [LinearGaugeAnnotation](#). The annotation can be rendered either by specifying the id of the element or specifying the code to create a new element that needs to be displayed in the gauge area.

<!-- markdownlint-disable MD036 -->

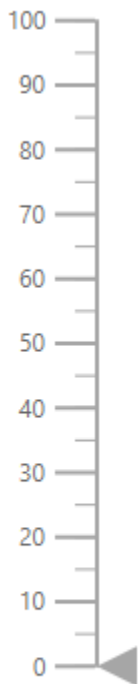
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Orientation(Orientation.Horizontal).Annotations(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "#walking",
        ZIndex = "1", X = 100, Y = 100
    },
}).Render()
<div>
    <script id="walking" type="text/x-template">
        <div id="annotation" style="width:100px;">
            
            </div>
        </script>
    </div>
```

ANNOTATIONS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Customization

The following properties are used to customize the annotation.

- [ZIndex](#) - Bring the annotation to the front or back, when annotation overlaps with another element.
- [AxisValue](#) - To place the annotation in the specified axis value with respect to the provided axis index.
- [AxisIndex](#) - To place the annotation in the specified axis with respect to the provided axis value.
- [HorizontalAlignment](#) - To place the annotation horizontally.

- [VerticalAlignment](#) - To place the annotation vertically.
- [X, Y](#) - To place the annotation in the specified location.

Changing the z-index

To change the stack order of an annotation element, the [ZIndex](#) property of the [LinearGaugeAnnotation](#) can be used.

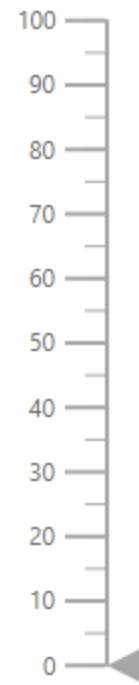
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Annotations(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "<div id='first'><h1>Gauge</h1></div>",
        ZIndex = "1"
    },
}).Render()
```

Z-ORDER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Gauge



Positioning an annotation

The annotation can be placed anywhere in the Linear Gauge by setting the pixel value to the [X](#), and [Y](#) properties in the [LinearGaugeAnnotation](#).

CSHTML

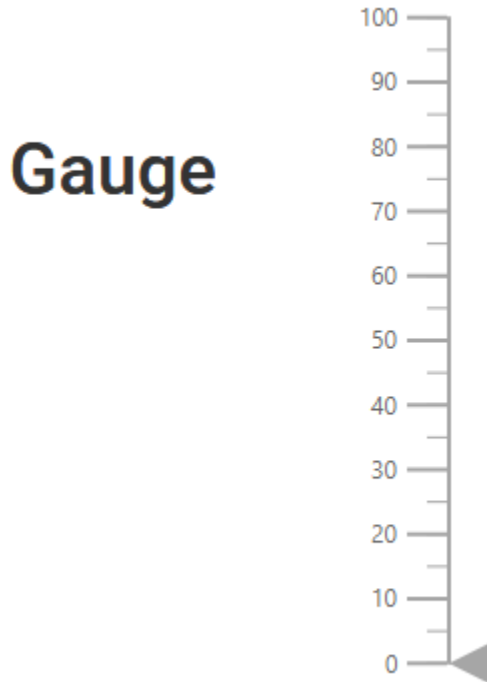
```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Annotations(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "<div id='first'><h1>Gauge</h1></div>",
        ZIndex = "1", X=250, Y=100
    },
}).Render()
```

ANIMATION-POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
```



```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```



<!-- markdownlint-disable MD036 -->

Alignment of annotation

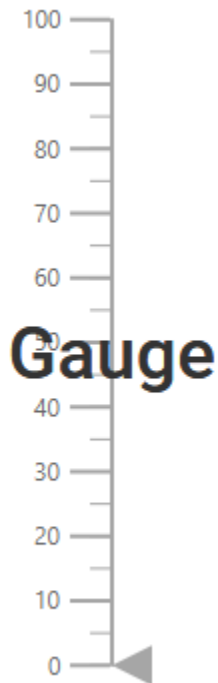
The annotation can be aligned horizontally and vertically by using the [HorizontalAlignment](#), and [VerticalAlignment](#) properties respectively. The possible values can be **Center**, **Far**, **Near**, and **None**. The [HorizontalAlignment](#), and [VerticalAlignment](#) are not applicable when the [X](#), and [Y](#) properties are set in the [LinearGaugeAnnotation](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Annotations(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "<div id='first'><h1>Gauge</h1></div>",
        ZIndex = "1", VerticalAlignment=Placement.Center,
        HorizontalAlignment=Placement.Center
    },
}).Render()
```

ANIMATION-ALIGNMENT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Multiple annotations

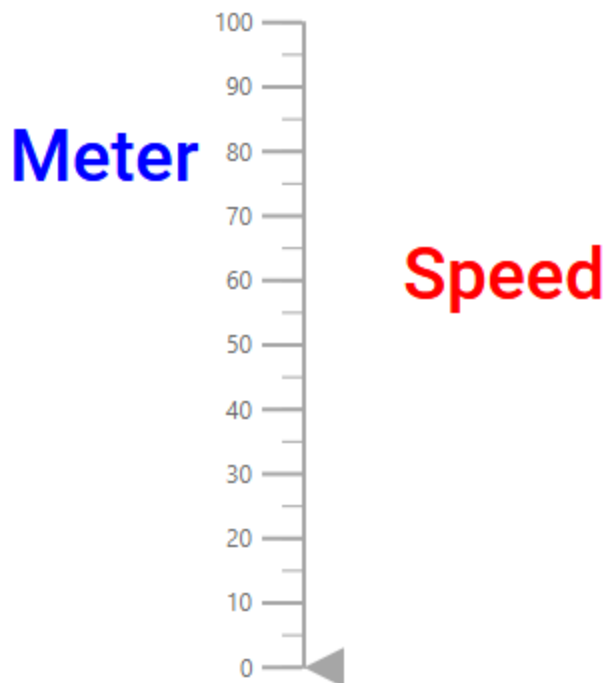
Multiple annotations can be added to the Linear Gauge component by adding the multiple [LinearGaugeAnnotation](#) in the [LinearGaugeAnnotations](#) and customization for the annotation can be done with the [LinearGaugeAnnotation](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Annotations(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "<div><h1 style='color: red;'>Speed</h1></div>",
        ZIndex = "1", VerticalAlignment=Placement.Near,
        HorizontalAlignment=Placement.Center, X=100, Y=150
    },
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "<div><h1 style='color: blue;'>Meter</h1></div>",
        ZIndex = "1", VerticalAlignment=Placement.Center,
        HorizontalAlignment=Placement.Center, X=-100, Y=-100
    }
}).Render()
```

MULTIPLE-ANNOTATIONS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Animation in ASP.NET MVC Linear Gauge

All of the elements in the Linear Gauge, such as the axis lines, ticks, labels, ranges, pointers, and annotations, can be animated sequentially by using the [AnimationDuration](#) property. The animation for the Linear Gauge is enabled when the [AnimationDuration](#) property is set to an appropriate value in milliseconds, providing a smooth rendering effect for the component. If the [AnimationDuration](#) property is set to **0**, which is the default value, the animation effect is disabled. If the animation is enabled, the component will behave in the following order.

1. The axis line, ticks, labels, and ranges will all be animated at the same time.
2. If available, pointers will be animated in the same way as [pointer animation](#).
3. If available, annotations will be animated.

The animation of the Linear Gauge is demonstrated in the following example.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("container").Load("gaugeLoad").Orientation(Syncfusion.EJ2.LinearGauge.Orientation.Horizontal).Annotations(
    annotation => annotation.AxisIndex(0).AxisValue(10).X(10).Y(-70).ZIndex("1").Content("<div style='width: 70px;margin-left:-3%;margin-top: 42%;font-size: 16px;'>10 MPH</div>").Add()).Axes(axes =>
    axes.Pointers(pointer =>
        pointer.Width(15).Height(15).Value(10).Offset("-40").Placement(Syncfusion.EJ2.LinearGauge.Placement.Near).MarkerType(Syncfusion.EJ2.LinearGauge.MarkerType.Triangle).Add())
```

```

        .MajorTicks(majorTick =>
majorTick.Interval(10).Height(20).Color("#9E9E9E"))
        .MinorTicks(minorTick =>
minorTick.Interval(2).Height(10).Color("#9E9E9E"))
        .Ranges(range =>
range.Start(0).End(50).StartWidth(10).EndWidth(10).Color("#F45656").Offset(3
5).Add())
        .LabelStyle(labelStyle => labelStyle.Offset(48).Font(font =>
font.FontFamily("inherit"))).Add()).Render()
<script>
    window.gaugeLoad = function (args) {
        args.gauge.animationDuration = 3000;
    }
</script>

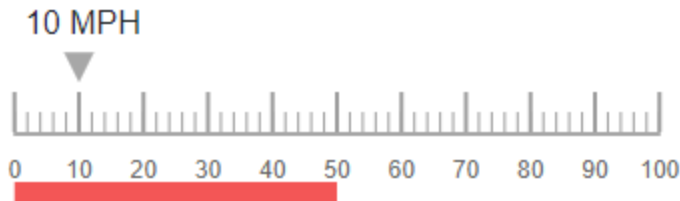
```

ANIMATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Only the pointer of the Linear Gauge can be animated individually, not the axis lines, ticks, labels, ranges, and annotations. You can refer this [link](#) to enable only pointer animation.

User Interaction in ASP.NET Linear Gauge

Tooltip

<!-- markdownlint-disable MD036 -->

Linear Gauge displays the details about a pointer value through [LinearGaugeTooltipSettings](#), when the mouse hovers over the pointer. To enable the tooltip, set [Enable](#) property as **true**.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Tooltip(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTooltipSettings
{
    Enable = true
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            }
        }
    }
}).Render()
```

TOOLTIP.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



<!-- markdownlint-disable MD013 -->

Tooltip format

<!-- markdownlint-disable MD013 -->

Tooltip in the Linear Gauge control can be formatted using the [Format](#) property in [LinearGaugeTooltipSettings](#). It is used to render the tooltip in certain format or to add a user-defined unit in the tooltip. By default, the tooltip shows the pointer value only. In addition to that, more information can be added in the tooltip. For example, the format **{value}km** shows pointer value with kilometer unit in the tooltip.

CSHTML

```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Tooltip(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTooltipSettings {
    Enable = true, Format = "{value}km"
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            }
        }
    }
}

```

```

    }
}
}) .Render ()

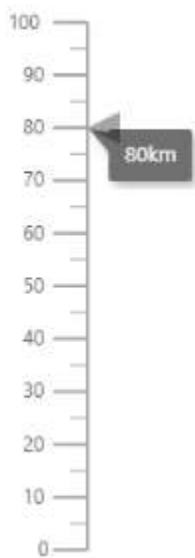
```

TOOLTIP-FORMAT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Tooltip Template

The HTML element can be rendered in the tooltip of the Linear Gauge using the [Template](#) property in [LinearGaugeTooltipSettings](#). The **{value}** can be used as placeholders in the HTML element to display the pointer values of the corresponding axis.

CSHTML


```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Tooltip(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTooltipSettings {
    Enable = true, Template = "<div>Pointer: 80 </div>"
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            }
        }
    }
}).Render()

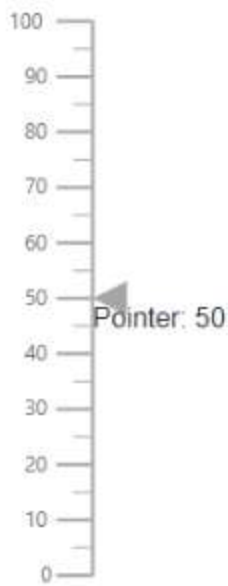
```

TOOLTIP-TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Customize the appearance of the tooltip

The tooltip can be customized using the following properties in [LinearGaugeTooltipSettings](#).

- [Fill](#) - To fill the color for tooltip.
- [EnableAnimation](#) - To enable or disable the tooltip animation.
- [Border](#) - To set the border color and width of the tooltip.
- [TextStyle](#) - To customize the style of the text in tooltip.
- [ShowAtMousePosition](#) - To show the tooltip at the mouse position.

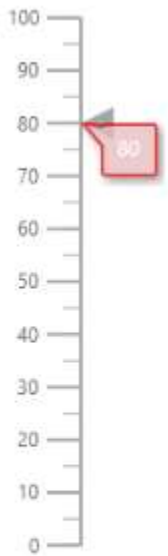
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Tooltip(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTooltipSettings {
    Enable = true, Fill = "#e5bcbc",
    Border = new Syncfusion.EJ2.LinearGauge.TooltipSettingsBorderTooltip
    {
        Width = 2, Color = "#d80000"
    }
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            }
        }
    }
})
```

```
}).Render()
```

TOOLTIP-APPEARANCE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Positioning the tooltip

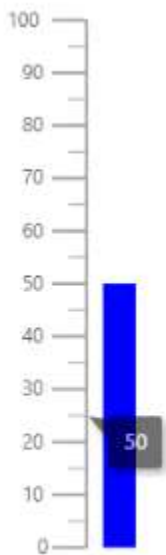
The tooltip is positioned at the **End** of the pointer. To change the position of the tooltip at the start, or center of the pointer, set the [Position](#) property to **Start** or **Center**.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Tooltip(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTooltipSettings {
    Enable = true, Position = TooltipPosition.Center
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 50, Type=Point.Bar, Color="blue"
            }
        }
    }
}).Render()
```

TOOLTIP-POSITION.CS

```
using Microsoft.AspNetCore.Mvc;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Pointer Drag

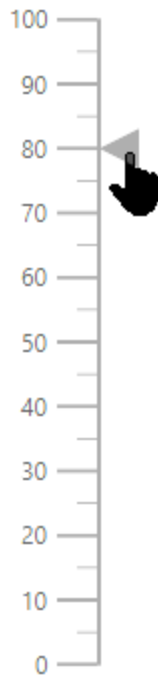
To drag either marker or bar pointer to the desired axis value, set the [EnableDrag](#) property as **true** in [LinearGaugePointer](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80, EnableDrag = true
            }
        }
    }
}).Render()
```

POINTER-DRAG.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Print and Export in ASP.NET MVC Linear Gauge

Print

The rendered Linear Gauge can be printed directly from the browser by calling the [print](#) method. To use the print functionality, set the [AllowPrint](#) property as **true**.

CSHTML

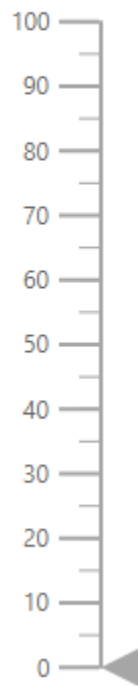
```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Print").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Height("450px").AllowPrint(true).Width("65
0px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.print();
    };
</script>
```

PRINT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
```

```
using Syncfusion.EJ2.LinearGauge;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            return View();  
        }  
    }  
}
```

PRINT



Export

Image Export

To use the image export functionality, set the [AllowImageExport](#) property as **true**. The rendered Linear Gauge can be exported as an image using the [export](#) method. This method requires two parameters: export type and file name. The Linear Gauge can be exported as an image with the following formats.

- JPEG
- PNG
- PDF

CSHTML

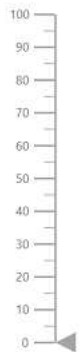
```
@using Syncfusion.EJ2;
```

```
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Height("450px").AllowImageExport(true).Wid
th("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.export('PNG', 'LinearGauge');
    };
</script>
```

EXPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Export



PDF Export

To use the PDF export functionality, set the [AllowPdfExport](#) property as **true**. The rendered Linear Gauge can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name, and orientation of the PDF document. The orientation of the PDF document can be set as **Portrait** or **Landscape**.

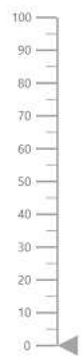
CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Height("450px").AllowPdfExport(true).Width
("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.export('PDF', 'LinearGauge', 0);
    };
</script>
```

EXPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Export

*Exporting Linear Gauge as base64 string of the file*

The Linear Gauge can be exported as base64 string for the JPEG, PNG and PDF formats. The rendered Linear Gauge can be exported as base64 string of the exported image or PDF document used in the

[export](#) method. The arguments that are required for this method is export type, file name, orientation of the exported PDF document and **allowDownload** boolean value that is set as **false** to return base64 string. The value for the orientation of the exported PDF document is set as **null** for image export and **Portrait** or **Landscape** for the PDF document.

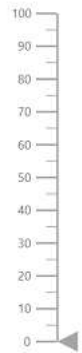
CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Height("450px").AllowImageExport(true).Width("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.export('JPEG', 'LinearGauge', null, false).then((data) =>
        {
            let base64 = data;
            document.writeln(base64);
        });
    };
</script>
```

EXPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Export



Note: The exporting of the Linear Gauge as base64 string is not applicable for the SVG format.

Appearance in ASP.NET MVC Linear Gauge

Customizing the Linear Gauge area

The following property and classes are available in the [LinearGauge](#) to customize the Linear Gauge area.

- [Background](#) - Applies the background color for the Linear Gauge.
- [Border](#) - To customize the color and width of the border in Linear Gauge.
- [Margin](#) - To customize the margins of the Linear Gauge.

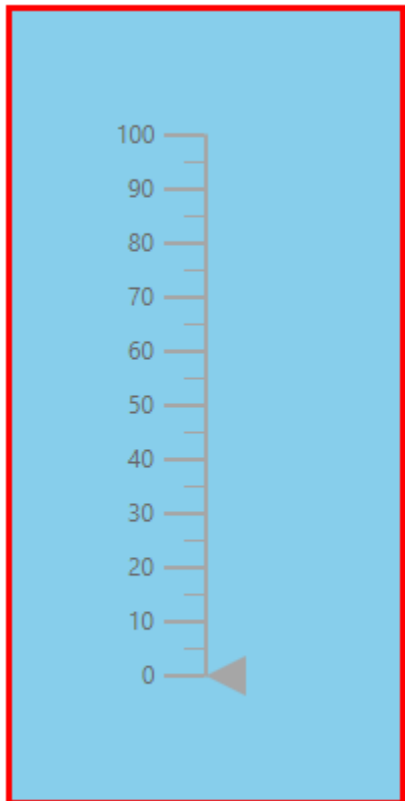
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Width("200px").Height("400px").Background("skyblue").Border(new Syncfusion.EJ2.LinearGauge.LinearGaugeBorder
{
    Width = 3, Color = "red"
}).Margin(new Syncfusion.EJ2.LinearGauge.LinearGaugeMargin {
    Left = 20, Bottom = 20, Right = 20, Top = 20
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
    }
}
).Render()
```

GAUGE-BACKGROUND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
```

```
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Setting up the Linear Gauge title

The title for the Linear Gauge can be set using [Title](#) property in Linear Gauge. Its appearance can be customized using the [TitleStyle](#) with the below properties.

- [Color](#) - Specifies the text color of the title.
- [FontFamily](#) - Specifies the font family of the title.
- [FontStyle](#) - Specifies the font style of the title.
- [FontWeight](#) - Specifies the font weight of the title.
- [Opacity](#) - Specifies the opacity of the title.
- [Size](#) - Specifies the font size of the title.

CSHTML

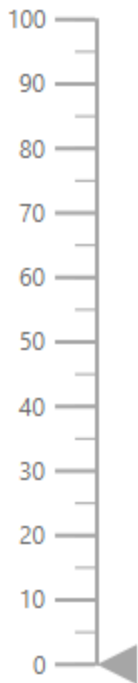
```
@using Syncfusion.EJ2.LinearGauge;
```

```
@Html.EJS().LinearGauge("gauge").Title("Linear Gauge").TitleStyle(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTitleStyleLinearGauge {
    FontFamily = "Arial",
    FontStyle = "italic",
    FontWeight = "regular",
    Color = "#E27F2D",
    Size = "23px",
    Opacity = 1
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
    }
})
).Render()
```

GAUGE-TITLE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Linear Gauge



Customizing the Linear Gauge container

The area used to render the ranges and pointers at the center position of the gauge is called container. The following types of container to be applicable for Linear Gauge.

- Normal
- Rounded Rectangle
- Thermometer

The type of the container can be modified by using the [Type](#) property in [LinearGaugeContainer](#). The container can be customized by using the following properties and class in [LinearGaugeContainer](#).

- [Offset](#) - To place the container with the specified distance from the axis of the Linear Gauge.
- [Width](#) - To set the thickness of the container.
- [Height](#) - To set the length of the container.
- [BackgroundColor](#) - To set the background color of the container.
- [Border](#) - To set the color and width for the border of the container.

Normal

The **Normal** type will render the container as a rectangle and this is the default container type.

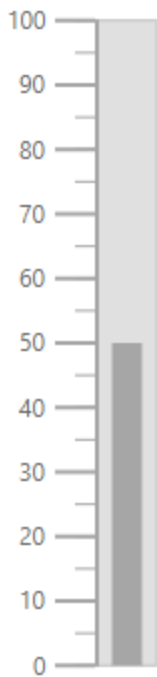
CSHTML

```
@using Syncfusion.EJ2.LinearGauge;  
@Html.EJS().LinearGauge("gauge").Container(new  
Syncfusion.EJ2.LinearGauge.LinearGaugeContainer  
{
```

```
        Width = 30
    }).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
    {
        new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
        {
            Pointers = new List<LinearGaugePointer>
            {
                new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
                {
                    Type = Point.Bar, Value = 50, Width = 15
                }
            }
        }
    })
    .Render()
```

GAUGE-CONTAINER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Rounded Rectangle

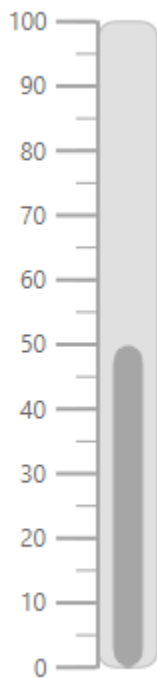
The **RoundedRectangle** type will render the container as a rectangle with rounded corner radius. The rounded corner radius of the container can be customized using the [RoundedCornerRadius](#) property in [LinearGaugeContainer](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Container(new
Syncfusion.EJ2.LinearGauge.LinearGaugeContainer
{
    Width = 30, Type= ContainerType.RoundedRectangle
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Type = Point.Bar, Value = 50, Width = 15
            }
        }
    }
}).Render()
```

ROUNDED-RECTANGLE.CS


```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Thermometer

The **Thermometer** type will render the container similar to the appearance of thermometer.

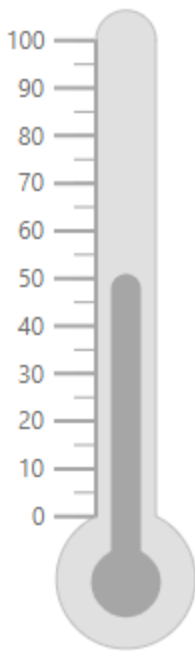
CHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").Container(new
Syncfusion.EJ2.LinearGauge.LinearGaugeContainer
{
```

```
Width = 30, Type= ContainerType.Thermometer
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Type = Point.Bar, Value = 50, Width = 15
            }
        }
    }
}).Render()
```

THERMO-METER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Fitting the Linear Gauge to the control

The Linear Gauge component is rendered with margin by default. To remove the margin around the Linear Gauge, the [AllowMargin](#) property in the [LinearGauge](#) is set as **false**.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").AllowMargin(false).Orientation(Orientation.
Horizontal).Height("300px").Background("skyblue").Border(new
Syncfusion.EJ2.LinearGauge.LinearGaugeBorder
{
    Width = 3, Color = "red"
}).Margin(new Syncfusion.EJ2.LinearGauge.LinearGaugeMargin {
    Left = 0, Bottom = 0, Right = 0, Top = 0
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
    }
}
).Render()
```

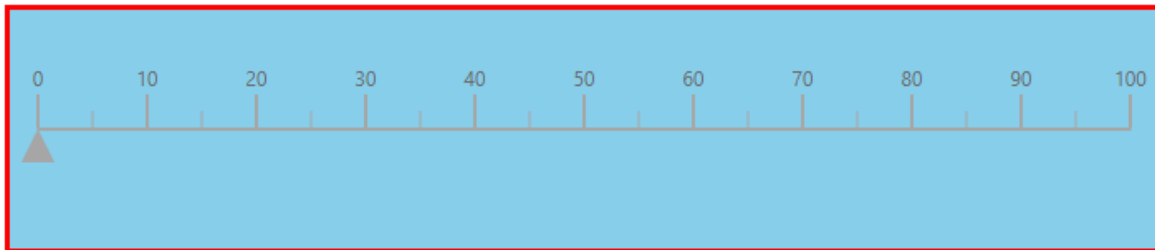
GAUGE-MARGIN.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
```

```

using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Note: To use this feature, set the [AllowMargin](#) property to **false**, the [Width](#) property to **100%** and the properties of [e-lineargauge-margin](#) to **0**.

Accessibility in ASP.NET MVC Linear Gauge

Linear Gauge has built-in accessibility features like screen reading and WAI-ARIA attributes.

WAI-ARIA attributes

The Linear Gauge component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Linear Gauge component:

| Attributes | Purpose |
|--------------------|---|
| --- | --- |
| role=region | It is specified in the pointer where the interactive drag and drop function is supported to update the pointer value. |
| aria-label | Provides an accessible name for the axis labels, text pointer and annotation. |

Screen reading in Linear Gauge

Accessibility in the Linear Gauge component ensures that all users, regardless of ability or disability, can use screen reading. The following Linear Gauge elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |
|--------------|--|
| --- | --- |
| Axis labels | Reads the axis labels of the Linear Gauge. |
| Text pointer | Reads the text content shown as a pointer in Linear Gauge. |

| Annotation | Reads the content specified in the annotation. |

Ensuring accessibility

The Linear Gauge component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Linear Gauge component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Linear Gauge component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Internationalization in ASP.NET MVC Linear Gauge

Globalization is the process of designing and developing a component that works in different cultures. Internationalization is used to globalize the number content in Linear Gauge component using [Format](#) property in Linear Gauge. It has static text on some features such as

- Axis label
- Tooltip

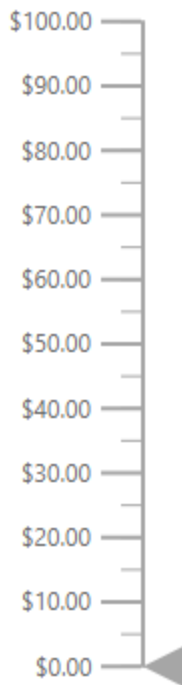
The static text on above features can be changed to any culture such as Arabic, Deutsch and French. To know more about the globalization in ASP.NET MVC components, refer [here](#).

Numeric Format

The text in axis labels and tooltip can be displayed in the numeric format such as currency, percentage and so on. To know more about the numeric formats in axis labels, refer [here](#). In the below example, the axis label is displayed in the currency format.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;  
@Html.EJS().LinearGauge("gauge").Format("c").Axes(new  
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>  
{  
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis  
    {  
    }  
}) .Render()
```



Events in ASP.NET MVC Linear Gauge

This section describes the Linear Gauge component's event that gets triggered when corresponding operations are performed.

AnimationComplete

When the pointer animation is completed, the [AnimationComplete](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").AnimationComplete("animationComplete").Axes(
    new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
    {
        new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
        {
            Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
            {
                new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
                {
                    Value = 10
                }
            }
        }
    }
).Render()
<script>
    function animationComplete(args) {
    }
```

```
</script>
```

AnnotationRender

Before the annotation is rendered in the Linear Gauge, the [AnnotationRender](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").AnnotationRender("annotationRender").Annotations(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "<div id='first'><h1>Gauge</h1></div>",
        ZIndex = "1",
        AxisValue = 0
    },
}).Render()
<script>
    function annotationRender(args) {
    }
</script>
```

AxisLabelRender

Before each axis label is rendered in the Linear Gauge, the [AxisLabelRender](#) event is fired. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").AxisLabelRender("axisLabelRender").Render()
<script>
    function axisLabelRender(args) {
    }
</script>
```

BeforePrint

The [BeforePrint](#) event is fired before the print begins. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").Content("Print").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Height("450px").AllowPrint(true).BeforePrint("beforePrint").Width("650px").Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.print();
    };

    function beforePrint(args) {
```

```

    }
</script>

```

DragEnd

The [DragEnd](#) event will be fired before the pointer drag is completed. To know more about the argument of this event, refer [here](#).

CSHTML

```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").DragEnd("dragEnd").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                EnableDrag = true
            }
        }
    }
}).Render()
<script>
    function dragEnd(args) {
    }
</script>

```

DragMove

The [DragMove](#) event will be fired when the pointer is dragged. To know more about the arguments of this event, refer [here](#).

CSHTML

```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").DragMove("dragMove").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                EnableDrag = true
            }
        }
    }
}).Render()
<script>
    function dragMove(args) {
    }
</script>

```


DragStart

When the pointer drag begins, the [DragStart](#) event is triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").DragStart("dragStart").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                EnableDrag = true
            }
        }
    }
}).Render()
<script>
    function dragStart(args) {
    }
</script>
```

GaugeMouseDown

When mouse is pressed down on the gauge, the [GaugeMouseDown](#) event is triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").GaugeMouseDown("gaugeMouseDown").Render()
<script>
    function gaugeMouseDown(args) {
    }
</script>
```

GaugeMouseLeave

When mouse pointer moves over the gauge, the [GaugeMouseLeave](#) event is triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").GaugeMouseLeave("gaugeMouseLeave").Render(
)
<script>
    function gaugeMouseLeave(args) {
    }
</script>
```

GaugeMouseMove

When mouse pointer leaves the gauge, the [GaugeMouseMove](#) event is triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").GaugeMouseMove("gaugeMouseMove").Render()
<script>
    function gaugeMouseMove(args) {
    }
</script>
```

GaugeMouseUp

When the mouse pointer is released over the Linear Gauge, the [GaugeMouseUp](#) event is triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").GaugeMouseUp("gaugeMouseUp").Render()
<script>
    function gaugeMouseUp(args) {
    }
</script>
```

Load

Before the Linear Gauge is loaded, the [Load](#) event is fired. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").Load("load").Render()
<script>
    function load(args) {
    }
</script>
```

Loaded

After the Linear Gauge has been loaded, the [Loaded](#) event will be triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").Loaded("loaded").Render()
<script>
    function loaded(args) {
    }
</script>
```

Resized

After the window resizing, the [Resized](#) event is triggered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").Resized("resized").Render()

<script>
    function resized(args) {
    }
</script>
```

TooltipRender

The [TooltipRender](#) event is fired before the tooltip is rendered. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("gauge").TooltipRender("tooltipRender").Tooltip(new
Syncfusion.EJ2.LinearGauge.LinearGaugeTooltipSettings
{
    Enable = true
}).Axes(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            }
        }
    }
}).Render()

<script>
    function tooltipRender(args) {
    }
</script>
```

ValueChange

The [ValueChange](#) event is triggered when the pointer is dragged from one value to another. To know more about the arguments of this event, refer [here](#).

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().LinearGauge("linear").ValueChange("valueChange").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
```

```

        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                EnableDrag = true
            }
        }
    }).Render()
<script>
    function valueChange(args) {
    }
</script>

```

Methods in ASP.NET MVC Linear Gauge

The following methods are available in the Linear Gauge component.

setPointerValue

To change the pointer value dynamically, use the [setPointerValue](#) method in the Linear Gauge component. The following are the arguments for this method.

| Argument name | Description |
|---------------|--|
| axisIndex | Specifies the index of the axis in which the pointer value is to be updated. |
| pointerIndex | Specifies the index of the pointer to be updated. |
| pointerValue | Specifies the value of the pointer to be updated. |

CSHTML

```

@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().Button("togglebtn").Content("Print").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 80
            }
        }
    }
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.setPointerValue(0, 0, 30);
    };
</script>

```

setAnnotationValue

To change the annotation content dynamically, use the [setAnnotationValue](#) method in the Linear Gauge component. The following are the arguments for this method.

| Argument name | Description |
|-----------------|---|
| annotationIndex | Specifies the index number of the annotation to be updated. |
| content | Specifies the text for the annotation to be updated. |
| axisValue | Specifies the value of the axis where the annotation is to be placed. |

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().Button("togglebtn").Content("Print").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
            {
                Value = 10
            }
        }
    }
}).Annotations(new List<Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation> {
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAnnotation
    {
        Content = "10", ZIndex = "1", AxisValue = 0
    }
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.setAnnotationValue(0, '50', 50);
    };
</script>
```

refresh

The [refresh](#) method can be used to change the state of the component and render it again.

CSHTML

```
@using Syncfusion.EJ2.LinearGauge;
@Html.EJS().Button("togglebtn").Content("Print").CssClass("e-flat").Render()
@Html.EJS().LinearGauge("linear").Axes(new
List<Syncfusion.EJ2.LinearGauge.LinearGaugeAxis>
{
    new Syncfusion.EJ2.LinearGauge.LinearGaugeAxis
    {
        Pointers = new List<Syncfusion.EJ2.LinearGauge.LinearGaugePointer>
        {
            new Syncfusion.EJ2.LinearGauge.LinearGaugePointer
```

```

        {
            Value = 10
        }
    }
}
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var linearObj = document.getElementById('linear').ej2_instances[0];
        linearObj.refresh();
    };
</script>

```

Migration from Essential JS 1

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

Linear gauge dimensions

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Height | **Property:** *height*

 @Html.EJ().LinearGauge("container")

 .Height("150px") | **Property:** *height*

 @Html.EJS().LinearGauge("container")

 .Height("150px").Render() |

| Width | **Property:** *width*

 @Html.EJ().LinearGauge("container")

 .Width("200px") | **Property:** *width*

 @Html.EJS().LinearGauge("container")

 .Width("200px").Render() |

| Height(In Percentage) | Not Applicable | **Property:** *height*

 @Html.EJS().LinearGauge("container")
 .Height("70%").Render() |

| Width(In Percentage) | Not Applicable | **Property:** *width*

 @Html.EJS().LinearGauge("container")
 .Height("100%").Render() |

Line customization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Height | **Property:** *scales.length*

 @Html.EJ().LinearGauge("container")
 .Scales(sc
 => {sc.Length(300).Add()}) | **Property:** *axes.line.height*

 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.Line(line =>
 line.Height(100)).Add())
 .Render() |

| Width | **Property:** *scales.width*

 @Html.EJ().LinearGauge("container")
 .Scales(sc =>
 {sc.Width(300).Add()}) | **Property:** *axes.line.width*

 @Html.EJS().LinearGauge("container")

 .Axes(axes => axes.Line(line => line.Width(50)).Add())
 .Render() |

| Color | **Property:** *scales.backgroundColor*

 @Html.EJ().LinearGauge("container")

 .Scales(sc => {sc.BackgroundColor("blue").Add()}) | **Property:** *axes.line.color*

@Html.EJS().LinearGauge("container")
 .Axes(axes => axes.Line(line => line.Color("Blue")).Add())
 .Render()|

| Offset | Not Applicable | **Property:** *axes.line.offset* @Html.EJS().LinearGauge("container") .Axes(axes => axes.Line(line => line.Offset(2)).Add()) .Render()|

| Opacity | **Property:** *scales.opacity* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Opacity(0.2).Add()})| Not Applicable |

| DashArray | Not Applicable | **Property:** *axes.line.dashArray* @Html.EJS().LinearGauge("container") .Axes(axes => axes.Line(line => line.DashArray(1)).Add()) .Render()|

Ticks

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Type of Ticks | **Property:** *scales.ticks.type* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval)}).Add() .Add()})| **Property:** *axes.majorTicks.height* @Html.EJS().LinearGauge("container") .Axes(axes => axes.MajorTicks().Add()) .Render()|

| Height of Major Ticks | **Property:** *scales.ticks.height* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval).Height(8).Add() }).Add()})| **Property:** *axes.majorTicks.height* @Html.EJS().LinearGauge("container") .Axes(axes => axes.MajorTicks(ma => ma.Height(8)).Add()) .Render()|

| Width of Major Ticks | **Property:** *scales.ticks.width* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval).Width(5).Add() }).Add()})| **Property:** *axes.majorTicks.width* @Html.EJS().LinearGauge("container") .Axes(axes => axes.MajorTicks(ma => ma.Width(5)).Add()) .Render()|

| Color of Major Ticks | **Property:** *scales.ticks.color* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval).Color("Blue").Add() }).Add()})| **Property:** *axes.majorTicks.color* @Html.EJS().LinearGauge("container") .Axes(axes => axes.MajorTicks(ma => ma.Color("Blue")).Add()) .Render()|

| Offset for Major Ticks | **Property:** *scales.ticks.distanceFromScale* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval) .distanceFromScale(5).Add(); }).Add()})| **Property:** *axes.majorTicks.offset* @Html.EJS().LinearGauge("container") .Axes(axes => axes.MajorTicks(ma => ma.Offset(2)).Add()) .Render()|

| Interval of Major Ticks | **Property:** *scales.majorIntervalValue* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.majorIntervalValue(15).Add()})| **Property:** *axes.majorTicks.interval* @Html.EJS().LinearGauge("container") .Axes(axes => axes.MajorTicks(ma => ma.Interval(15)).Add()) .Render()|

| Angle of Major Ticks | **Property:** *scales.ticks.angle*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval).Angle(30).Add();
 }).Add()}) | Not Applicable |

| Opacity of Major Ticks | **Property:** *scales.ticks.opacity*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MajorInterval).Opacity(0.2).Add();
 }).Add()}) | Not Applicable |

| Height of Minor Ticks | **Property:** *scales.ticks.height*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MinorInterval).Height(8).Add();
 }).Add()}) | **Property:** *axes.minorTicks.height*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Height(8)).Add();
 .Render() |

| Width of Minor Ticks | **Property:** *scales.ticks.width*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MinorInterval).Width(5).Add();
 }).Add()}) | **Property:** *axes.minorTicks.width*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Width(5)).Add();
 .Render() |

| Color of Minor Ticks | **Property:** *scales.ticks.color*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MinorInterval).Color("Blue").Add();
 }).Add()}) | **Property:** *axes.minorTicks.color*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Color("Blue")).Add();
 .Render() |

| Offset for Minor Ticks | **Property:** *scales.ticks.distanceFromScale*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MinorInterval).distanceFromScale(5).Add();
 }).Add()}) | **Property:** *axes.minorTicks.offset*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Offset(2)).Add();
 .Render() |

| Interval of Minor Ticks | **Property:** *scales.minorIntervalValue*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.minorIntervalValue(15).Add()}) | **Property:** *axes.minorTicks.interval*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.MinorTicks(mi => mi.Interval(15)).Add();
 .Render() |

| Angle of Minor Ticks | **Property:** *scales.ticks.angle*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MinorInterval).Angle(30).Add();
 }).Add()}) | Not Applicable |

| Opacity of Minor Ticks | **Property:** *scales.ticks.opacity*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ticks(tic => {tic.Type(TickType.MinorInterval).Opacity(0.2).Add();
 }).Add()}) | Not Applicable |

Labels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Angle | **Property:** *scales.labels.angle*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Labels(lab => {lab.Angle(15).Add();
 }).Add()}) | Not Applicable |

| Offset | **Property:** *scales.labels.distanceFromScale* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.distanceFromScale(dis => {dis.x(15).Add();}).Add(); }).Add();}) | **Property:** *axes.labelStyle.offset* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.Offset(2)).Add()) .Render() |

| Format | **Property:** *scales.labels.unitText* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.UnitText("F").Add(); }).Add();}) | **Property:** *axes.labelStyle.format* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.Format("C")).Add()) .Render() |

| Unit Text Placement | **Property:** *scales.labels.unitTextPlacement* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.UnitTextPlacement("Front").Add(); }).Add();}) | Not Applicable |

| Label Range Color | Not Applicable | **Property:** *axes.labelStyle.useRangeColor* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.UseRangeColor(true)).Add()).Render() |

| Opacity | **Property:** *scales.labels.opacity* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.Opacity(0.5) .Add();}).Add();}) | **Property:** *axes.labelStyle.font.opacity* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.Font("fontstyle")) .Add()).Render() var fontstyle = new { opacity = 0.5 }; |

| Label Text Color | **Property:** *scales.labels.textColor* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.TextColor("red") .Add();}).Add();}) | **Property:** *axes.labelStyle.font.color* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.Font("fontstyle")) .Add()).Render() var fontstyle = new { color = 'red' }; |

| Label Font Family | **Property:** *scales.labels.font.fontFamily* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.Font(fon => {fon.FontFamily("SegoeUI").Add();}).Add(); }).Add();}) | **Property:** *axes.labelStyle.font.fontFamily* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.Font("fontstyle")) .Add()).Render() var fontstyle = new { fontFamily: 'Arial' }; |

| Label Font Style | **Property:** *scales.labels.font.fontStyle* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Labels(lab => {lab.Font(fon => {fon.FontStyle("Normal").Add();}).Add(); }).Add();}) | **Property:** *axes.labelStyle.font.fontStyle* @Html.EJS().LinearGauge("container") .Axes(axes => axes.LabelStyle(lab => lab.Font("fontstyle")) .Add()).Render() var fontstyle = new { fontStyle = 'Bold' }; |

| Label Size | **Property:** *scales.labels.font.size*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Labels(lab => {lab.Font(fon => {fon.Size("20px").Add();}).Add();
 }).Add();}) | **Property:** *axes.labelStyle.font.size*

 @Html.EJS().LinearGauge("container")

```
<br/> .Axes(axes => axes.LabelStyle(lab => lab.Font("fontstyle"))) <br/> .Add()).Render() <br/> var fontstyle = new { size= "15px" };|
```

```
|Label Font Weight| Not Applicable| Property: axes.labelStyle.font.fontWeight <br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.LabelStyle(lab =>
lab.Font("fontstyle"))) <br/> .Add()).Render() <br/> var fontstyle = new { fontWeight= '4' };|
```

Axis

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
|Minimum Value| Property: scales.minimum<br/><br/> @Html.EJ().LinearGauge("container") <br/>
.Scales(sc => {sc.Minimum(20).Add()})| Property: axes.minimum<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.Minimum(20).Add()).Render()|
```

```
|Maximum Value| Property: scales.maximum<br/><br/> @Html.EJ().LinearGauge("container") <br/>
.Scales(sc => {sc.Maximum(20).Add()})| Property: axes.maximum<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.Maximum(20).Add()).Render()|
```

```
|Inverted Position| Not Applicable| Property: axes.isInversed<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.IsInversed(true).Add()).Render()|
```

```
|Opposed Position| Not Applicable| Property: axes.opposedPosition<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.OpposedPosition(true).Add()).Render()|
```

Ranges

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
|Start Value| Property: scales.ranges.startValue<br/><br/> @Html.EJ().LinearGauge("container")
<br/> .Scales(sc => {sc.Ranges(ran => {ran.StartValue(20) <br/> .Add();}).Add()})| Property:
axes.ranges.start<br/><br/> @Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.Ranges(ran => ran.Start(20)) <br/> .Add()).Render()|
```

```
|End Value| Property: scales.ranges.endValue<br/><br/> @Html.EJ().LinearGauge("container")
<br/> .Scales(sc => {sc.Ranges(ran => {ran.EndValue(20) <br/> .Add();}).Add()})| Property:
axes.ranges.end<br/><br/> @Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.Ranges(ran => ran.End(20)) <br/> .Add()).Render()|
```

```
|Start Width| Property: scales.ranges.startWidth<br/><br/> @Html.EJ().LinearGauge("container")
<br/> .Scales(sc => {sc.Ranges(ran => {ran.StartWidth(20) <br/> .Add();}).Add()})| Property:
axes.ranges.startWidth<br/><br/> @Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.Ranges(ran => ran.StartWidth(20)) <br/> .Add()).Render()|
```

```
|End Width| Property: scales.ranges.endWidth <br/><br/> @Html.EJ().LinearGauge("container")
<br/> .Scales(sc => {sc.Ranges(ran => {ran.EndWidth(20) <br/> .Add();}).Add()})| Property:
axes.ranges.endWidth<br/><br/> @Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.Ranges(ran => ran.EndWidth(20)) <br/> .Add()).Render()|
```

|Color| **Property:** *scales.ranges.backgroundColor* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(ran => {ran.BackgroundColor("Red") .Add();}).Add();})|
Property: *axes.ranges.color*

 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.Color("red")))
 .Add()).Render()|

|Offset| **Property:** *scales.ranges.distanceFromScale* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(ran => {ran.DistanceFromScale(10) .Add();}).Add();})|
Property: *axes.ranges.offset*

 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.Ranges(ran => ran.Offset(5))
 .Add()).Render()|

|Range Position| **Property:** *scales.ranges.placement* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(ran => {ran.Placement("Near") .Add();}).Add();})| **Property:** *axes.ranges.position* @Html.EJS().LinearGauge("container") .Axes(axes => axes.Ranges(ran => ran.Position("Inside")) .Add()).Render()|

|Opacity| **Property:** *scales.ranges.opacity* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(ran => {ran.Opacity(0.5) .Add();}).Add();})| Not Applicable|

|Border Customization| **Property:** *scales.ranges.border* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(ran => {ran.Border(bor => {bor.Color("green") .Width(2).Add();}).Add();}).Add();})| **Property:** *axes.ranges.border* @Html.EJS().LinearGauge("container") .Axes(axes => axes.Ranges(ran => ran.Border(border)) .Add()).Render() var border = new { color='green', width="2"};|

Bar Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Bar Pointer| **Property:** *scales.ranges.barPointers.value* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(BarPointers => {br.Value(20).Add();}).Add();})| **Property:** *axes.pointers.value* @Html.EJS().LinearGauge("container") .Axes(axes => axes.pointers(po => po.Value(20) .Type("RangeBar")).Add()).Render()|

|Color of Bar Pointer| **Property:** *scales.ranges.barPointers.backgroundColor* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(BarPointers => {br.Value(20) .backgroundColor("red").Add();}).Add();})| **Property:** *axes.pointers.color* @Html.EJS().LinearGauge("container") .Axes(axes => axes.pointers(po => po.Value(20) .Type("RangeBar").Color("Red")).Add()).Render()|

|Offset of Bar Pointer| **Property:** *scales.ranges.barPointers.distanceFromScale* @Html.EJ().LinearGauge("container") .Scales(sc => {sc.Ranges(BarPointers => {br.distanceFromScale(20) .Add();}).Add();})| **Property:** *axes.pointers.offset* @Html.EJS().LinearGauge("container") .Axes(axes => axes.pointers(po => po.Value(20) .Type("RangeBar").Offset(20)).Add()).Render()|

| Opacity of Bar Pointer | **Property:** *scales.ranges.barPointers.opacity*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ranges(BarPointers => {br.Opacity(0.5)
 .Add();}).Add();}) | **Property:** *axes.pointers.opacity*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.pointers(po => po.Value(20)
 .Type("RangeBar").Opacity(0.5)).Add()).Render() |

| Width of Bar Pointer | **Property:** *scales.ranges.barPointers.width*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ranges(BarPointers => {br.Width(2)
 .Add();}).Add();}) | **Property:** *axes.pointers.width*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.pointers(po => po.Value(20)
 .Type("RangeBar").Width(2)).Add()).Render() |

| Gradients of Bar Pointer | **Property:** *scales.ranges.barPointers.gradients*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ranges(BarPointers => {
 br.gradients(gra => {gra.ColorStop(0).Color("#FFFFFF").Add();})
 .Add();}).Add();}) | Not Applicable |

| Border of Bar Pointer | **Property:** *scales.ranges.barPointers.border*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ranges(BarPointers => {br.Border(bor => {bor.Color("red").Width(2).Add();})
 .Add();}).Add();}) | **Property:** *axes.pointers.border*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.pointers(po => po.Value(20)
 .Type("RangeBar").Border(border)).Add()).Render()
 var border = new { color="red" width="2" }; |

| Animation of Bar Pointer | **Property:** *enableAnimation*
 @Html.EJ().LinearGauge("container")
 .EnableAnimation(true) | **Property:** *axes.pointers.animationDuration*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.pointers(po => po.Value(20)
 .Type("RangeBar").animationDuration(1000)).Add()).Render() |

| Rounded Corner of Bar Pointer | Not Applicable | **Property:** *axes.pointers.roundedCornerRadius*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.pointers(po => po.Value(20)
 .Type("RangeBar").RoundedCornerRadius(10)).Add()).Render() |

Marker Pointer

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Pointer | **Property:** *scales.ranges.markerPointers.value*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ranges(MarkerPointers => {mr.Value(20).Add();}).Add();}) | **Property:** *axes.pointers.value*
 @Html.EJS().LinearGauge("container")
 .Axes(axes => axes.pointers(po => po.Value(20)
 .Type("Marker")).Add()).Render() |

| Color of Marker Pointer | **Property:** *scales.ranges.markerPointers.backgroundColor*
 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.Ranges(MarkerPointers => {mr.Value(20).Color("Red").Add();}).Add();}) | **Property:** *axes.pointers.color*

```
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").Color("Red")).Add()).Render()|
```

| Offset of Marker Pointer | **Property:** *scales.ranges.markerPointers
.distanceFromScale

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers =>
{mr.Value(20).distanceFromScale(10).Add();}) <br/> .Add();})| Property:
axes.pointers.offset<br/><br/> @Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.pointers(po => po.Value(20) <br/> .Type("Marker").Offset(20)).Add()).Render()|
```

| Opacity of Marker Pointer | **Property:** *scales.ranges.markerPointers.opacity

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers =>
{mr.Opacity(0.5) <br/> .Add();}).Add();})| Property: axes.pointers.opacity<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").Opacity(0.5)).Add()).Render()|
```

| Width of Marker Pointer | **Property:** *scales.ranges.markerPointers.width

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers =>
{mr.Width(2) <br/> .Add();}).Add();})| Property: axes.pointers.width<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").Width(2)).Add()).Render()|
```

| Gradients of Marker Pointer | **Property:** *scales.ranges.markerPointers
.gradients

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers => { <br/>
mr.gradients(gra => {gra.ColorStop(0).Color("#FFFFFF").Add();}) <br/> .Add();}).Add();})| Not
Applicable|
```

| Border of Marker Pointer | **Property:** *scales.ranges.markerPointers.border

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers =>
{mr.Border(bor => {bor.Color("red").Width(2).Add();}) <br/> .Add();}).Add();})| Property:
axes.pointers.border<br/><br/> @Html.EJS().LinearGauge("container") <br/> .Axes(axes =>
axes.pointers(po => po.Value(20) <br/> .Type("Marker").Border(border)).Add()).Render() <br/>
var border = new { color="red" width="2" };|
```

| Animation of Marker Pointer | **Property:** *enableMarkerPointerAnimation

*

```
@Html.EJ().LinearGauge("container") <br/> .EnableMarkerPointerAnimation(true)| Property:
axes.pointers.animationDuration<br/><br/> @Html.EJS().LinearGauge("container") <br/>
.Axes(axes => axes.pointers(po => po.Value(20) <br/>
.Type("Marker").animationDuration(1000)).Add()).Render()|
```

| Type of Marker Pointer | **Property:** *scales.ranges.markerPointers.type

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers =>
{mr.Type("Diamond") <br/> .Add();}).Add();})| Property: axes.pointers.markerType<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").MarkerType("Diamond")).Add()).Render()|
```

| Placement of Marker Pointer | **Property:** *scales.ranges.markerPointers
.placement

*

```
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.Ranges(MarkerPointers =>
{mr.Placement("Near") <br/> .Add();}).Add();})| Property: axes.pointers.placement<br/><br/>
```

```
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").Placement("Near")).Add()).Render()|
```

```
| Drag of Marker Pointer | Property: readOnly<br/><br/> @Html.EJ().LinearGauge("container") <br/>
.ReadOnly(true)| Property: axes.pointers.enableDrag<br/><br/>
```

```
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").EnableDrag(true)).Add()).Render()|
```

```
| Image Marker Pointer | Not Applicable| Property: axes.pointers.imageUrl<br/><br/>
```

```
@Html.EJS().LinearGauge("container") <br/> .Axes(axes => axes.pointers(po => po.Value(20)
<br/> .Type("Marker").ImageUrl("")).Add()).Render()|
```

Annotation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```
| Content| Property: scales.customLabels.value<br/><br/> @Html.EJ().LinearGauge("container")
<br/> .Scales(sc => {sc.customLabels(cus => {cus.Value("Linear Gauge") <br/> .Add();}).Add();}).Add();}|
Property: annotations.content<br/><br/> @Html.EJS().LinearGauge("container") <br/>
.Annotations(ann => ann.Content("Linear Gauge").Add()).Render()|
```

```
| Horizontal Alignment| Not Applicable| Property: annotations.horizontalAlignment<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann =>
ann.HorizontalAlignment("Center").Add()).Render()|
```

```
| Vertical Alignment| Not Applicable| Property: annotations.verticalAlignment<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann =>
ann.VerticalAlignment("Center").Add()).Render()|
```

```
| Position of X| Property: scales.customLabels.position.x<br/><br/>
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.customLabels(cus =>
{cus.Position(po => {po.X(20).Add();}) <br/> .Add();}).Add();})| Property: annotations.x<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann => ann.X(35).Add()).Render()|
```

```
| Position of Y| Property: scales.customLabels.position.y<br/><br/>
@Html.EJ().LinearGauge("container") <br/> .Scales(sc => {sc.customLabels(cus =>
{cus.Position(po => {po.Y(20).Add();}) <br/> .Add();}).Add();})| Property: annotations.y<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann => ann.Y(35).Add()).Render()|
```

```
| Z Index| Not Applicable| Property: annotations.zIndex<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann =>
ann.ZIndex("1").Add()).Render()|
```

```
| Axis Index| Not Applicable| Property: annotations.axisIndex<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann =>
ann.AxisIndex("0").Add()).Render()|
```

```
| Axis Value| Not Applicable| Property: annotations.axisValue<br/><br/>
@Html.EJS().LinearGauge("container") <br/> .Annotations(ann =>
ann.AxisValue("35").Add()).Render()|
```


| Font customization | **Property:** *scales.customLabels.font*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.customLabels(cus => {cus.Font(fo
 => {fo.Size("20px").Add();})
 .Add();})
 .Add();}) | **Property:** *annotations.font*

 @Html.EJS().LinearGauge("container")
 .Annotations(ann =>
 ann.Font(font).Add()).Render()
 var font =new {size='12px'}|

| Annotation Color | **Property:** *scales.customLabels.color*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.customLabels(cus =>
 {cus.Color("Red")
 .Add();})
 .Add();}) | **Property:** *annotations.font.color*

 @Html.EJS().LinearGauge("container")
 .Annotations(ann =>
 ann.Font(font).Add()).Render()
 var font =new {color = "red"}|

| Opacity of Annotation | **Property:** *scales.customLabels.opacity*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.customLabels(cus =>
 {cus.Opacity(0.4)
 .Add();})
 .Add();}) | **Property:** *annotations.font.opacity*

 @Html.EJS().LinearGauge("container")
 .Annotations(ann =>
 ann.Font(font).Add()).Render()
 var font =new {opacity = "0.4"}|

| Position Type | **Property:** *scales.customLabels.positionType*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.customLabels(cus =>
 {cus.PositionType("Outer")
 .Add();})
 .Add();}) | Not applicable|

| TextAngle of Annotation | **Property:** *scales.customLabels.textAngle*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.customLabels(cus =>
 {cus.TextAngle(25)
 .Add();})
 .Add();}) | Not applicable|

Tooltip

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Tooltip for Pointer | Not Applicable | **Property:** *tooltip.enable*

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool => tool.Enable(true).Add()).Render()|

| Tooltip for Label | **Property:** *tooltip.showLabelTooltip*

 @Html.EJ().LinearGauge("container")
 .Scales(sc => {sc.tooltip(tool =>
 {tool.ShowLabelTooltip(true)
 .Add();})
 .Add();}) | Not Applicable|

| Tooltip Format | Not Applicable | **Property:** *tooltip.format*

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool =>
 tool.Format({pointers.value}).Add()).Render()|

| Tooltip Color | Not Applicable | **Property:** *tooltip.fill*

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool => tool.Fill("Gray").Add()).Render()|

| Tooltip Template | **Property:** *tooltip.templateID*

 @Html.EJ().LinearGauge("container")

 .Scales(sc => {sc.tooltip(tool => {tool.TemplateId(true)
 .Add();})
 .Add();}) | **Property:**
tooltip.template

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool =>
 tool.Template("Temaplate").Add()).Render()|

| Tooltip Animation | Not Applicable | **Property:** *tooltip.enableAnimation*

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool =>
 tool.EnableAnimation(true).Add()).Render() |

| Tooltip Border | Not Applicable | **Property:** *tooltip.border*

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool =>
 tool.Border(border).Add()).Render()
 var border = new {width: 2, color: 'red'} |

| Tooltip TextStyle | Not Applicable | **Property:** *tooltip.textStyle*

 @Html.EJS().LinearGauge("container")
 .Tooltip(tool =>
 tool.TextStyle("Bold").Add()).Render() |

Appearance of Linear Gauge

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Background Color | **Property:** *backgroundColor*

 @Html.EJ().LinearGauge("container")

 .BackgroundColor("red") | **Property:** *background*

 @Html.EJS().LinearGauge("container")
 .Background("Red").Render() |

| Border Color | **Property:** *borderColor*

 @Html.EJ().LinearGauge("container")

 .BorderColor("blue") | **Property:** *border.color*

 @Html.EJS().LinearGauge("container")

 .Border(border).Render()
 var border= new {color="red"} |

| Margin | Not Applicable | **Property:** *margin*

 @Html.EJS().LinearGauge("container")

 .Margin(margin).Render()
 var margin= new {left: 40, right: 40, top: 40, bottom: 40} |

| Orientation | **Property:** *orientation*

 @Html.EJ().LinearGauge("container")

 .Orientation("Vertical") | **Property:** *orientation*

 @Html.EJS().LinearGauge("container")

 .Orientation("Vertical").Render() |

| Locale | **Property:** *locale* @Html.EJ().LinearGauge("container") .Locale("en-US") |
Property: *locale*

 @Html.EJS().LinearGauge("container")
 .Locale("en-
 US").Render() |

| Theme | **Property:** *theme*

 @Html.EJ().LinearGauge("container")

 .Theme("saffron") | **Property:** *theme*

 @Html.EJS().LinearGauge("container")

 .Theme("Material").Render() |

| Gauge Title | Not Applicable | **Property:** *title*

 @Html.EJS().LinearGauge("container")

 .Title("LinearGauge").Render() |

Gauge Container type

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Container Type | **Property:** *scales.type*

 @Html.EJ().LinearGauge("container")

 .Scales(sc => {sc.Type("Thermometer").Add()}) | **Property:** *container.type*

 @Html.EJS().LinearGauge("container")
 .Container(con =>
 con.Type("Thermometer").Add())
 .Render() |

| Container Height | Not Applicable | **Property:** *container.height*

 @Html.EJS().LinearGauge("container")
 .Container(con =>
 con.Type("Thermometer").Height(20).Add())
 .Render() |

| Container Width | Not Applicable | **Property:** *container.width*

 @Html.EJS().LinearGauge("container")
 .Container(con =>
 con.Type("Thermometer").Width(10).Add())
 .Render() |

| Container Offset | Not Applicable | **Property:** *container.offset*

 @Html.EJS().LinearGauge("container")
 .Container(con =>
 con.Type("Thermometer").Offset(5).Add())
 .Render() |

Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Animation Complete Event | Not Applicable | **Event:** *animationComplete*

 @Html.EJS().LinearGauge("container")

 .AnimationComplete("animationComplete").Render()
 function animationComplete(args)
 {} |

| Annotation Render Event | **Event:** *drawCustomLabel*

 @Html.EJ().CircularGauge("container")
 .DrawCustomLabel("customLabel")
 function
 customLabel(args) {} | **Event:** *annotationRender*

 @Html.EJS().LinearGauge("container")

 .AnnotationRender("annotationRender").Render()
 function annotationRender(args)
 {} |

| AxisLabel Render Event | **Event:** *drawLabels*

 @Html.EJ().CircularGauge("container")

 .DrawLabels("drawLabels")
 function drawLabels(args) {} | **Event:**
axisLabelRender

 @Html.EJS().LinearGauge("container")

 .AxisLabelRender("axisLabelRender").Render()
 function axisLabelRender(args) {} |

| Load Event | **Event:** *load*

 @Html.EJ().CircularGauge("container")

 .Load("load")
 function load(args) {} | **Event:** *load*

 @Html.EJS().LinearGauge("container")
 .Load("load").Render()
 function load(args) {} |

| Loaded Event | Not Applicable | **Event:** *loaded*

 @Html.EJS().LinearGauge("container")

 .Loaded("loaded").Render()
 function loaded(args) {} |

| Resize Event | Not Applicable | **Event:** *resized*

 @Html.EJS().LinearGauge("container")

 .Resized("resized").Render()
 function resized(args) {} |

| Tooltip Render Event | Not Applicable | **Event:** *tooltipRender*

 @Html.EJS().LinearGauge("container")
 .TooltipRender("tooltipRender").Render()

 function tooltipRender(args) {} |

| Value Change Event | Not Applicable | **Event:** *valueChange*

 @Html.EJS().LinearGauge("container")
 .ValueChange("valueChange").Render()

 function valueChange(args) {} |

| Mouse Move Event | **Event:** *mouseClickMove*

 @Html.EJ().CircularGauge("container")

 .MouseClickedMove("mouseClickMove")
 function mouseClickedMove(args) {} | **Event:**
gaugeMouseMove

 @Html.EJS().LinearGauge("container")

 .GaugeMouseMove("gaugeMouseMove").Render()
 function gaugeMouseMove(args) {} |

| Mouse Up Event | **Event:** *mouseClickUp*

 @Html.EJ().CircularGauge("container")

 .MouseClickedUp("mouseClickUp")
 function mouseClickedUp(args) {} | **Event:**
gaugeMouseUp

 @Html.EJS().LinearGauge("container")

 .GaugeMouseUp("gaugeMouseUp").Render()
 function gaugeMouseUp(args) {} |

| Mouse Down Event | Not Applicable | **Event:** *gaugeMouseDown*

 @Html.EJS().LinearGauge("container")

 .GaugeMouseDown("gaugeMouseDown").Render()
 function gaugeMouseDown(args) {} |

| Mouse Leave Event | Not Applicable | **Event:** *gaugeMouseLeave*

 @Html.EJS().LinearGauge("container")

 .GaugeMouseLeave("gaugeMouseLeave").Render()
 function gaugeMouseLeave(args) {} |

| Mouse Click Event | **Event:** *mouseClick*

 @Html.EJ().CircularGauge("container")

 .MouseClicked("mouseClick")
 function mouseClicked(args) {} | Not Applicable |

| Render Complete Event | **Event:** *renderComplete*

 @Html.EJ().CircularGauge("container")

 .RenderComplete("renderComplete")
 function renderComplete(args) {} | Not
 Applicable |

| Double Click Event | **Event:** *doubleClick*

 @Html.EJ().CircularGauge("container")

 .DoubleClick("doubleClick")
 function doubleClick(args) {} | Not Applicable |

| Right Click Event | **Event:** *rightClick*

 @Html.EJ().CircularGauge("container")

 .RightClick("rightClick")
 function rightClick(args) {} | Not Applicable |

| BarPointers Event | **Event:** *drawBarPointers*

 @Html.EJ().CircularGauge("container")

 .DrawBarPointers("drawBarPointers")
 function drawBarPointers(args) {} | Not
 Applicable |

| Indicators Event | **Event:** *drawIndicators*

 @Html.EJ().CircularGauge("container")

 .DrawIndicators("drawIndicators")
 function drawIndicators(args) {} | Not Applicable |

| MarkerPointer Event | **Event:** *drawMarkerPointers*

 @Html.EJ().CircularGauge("container")

 .DrawMarkerPointers("drawMarkerPointers")
 function drawMarkerPointers(args) {} | Not
 Applicable |

| Ranges Event | **Event:** *drawRange*

 @Html.EJ().CircularGauge("container")

 .DrawRange("drawRange")
 function drawRange(args) {} | Not Applicable |

| Gauge Initialized Event | **Event:** *init*

 @Html.EJ().CircularGauge("container")

 .Init("init")
 function init(args) {} | Not Applicable |

How To

<!-- markdownlint-disable MD034 -->

<!-- markdownlint-disable MD036 -->

Rendering linear gauges dynamically

In the linear gauge control, you can render the number of gauges dynamically and update the pointer values in ajax call.

To render the number of linear gauges dynamically, follow the given steps:

Step 1:

Initialize the data table collection with stock details.

```
`cs
public ActionResult Index()
{
    DataTable dt = new DataTable();
    dt.Columns.Add("Name");
    dt.Columns.Add("FullName");
    dt.Columns.Add("Stock");
    dt.Columns.Add("MaxStock");
    dt.Columns.Add("FreeStock");
    dt.Columns.Add("AvailableStock");
    dt.Columns.Add("StockCode");
    dt.Columns.Add("Status");
    dt.Rows.Add(new Object[] { "T13", "PM01 T13", 5000, 10000, 5000, 8230, "F" });
    dt.Rows.Add(new Object[] { "T14", "PM01 T14", 4000, 9000, 5000, 7230, "F" });
    dt.Rows.Add(new Object[] { "T15", "PM01 T15", 3000, 8000, 5000, 5230, "F" });
    dt.Rows.Add(new Object[] { "T16", "PM01 T16", 2000, 7000, 5000, 5230, "F" });
    List<LinearData> gauges = new List<LinearData>();
    foreach (DataRow row in dt.Rows)
    {
        gauges.Add(CreateLinearGauge(row));
    }
    ViewBag.GaugeData = gauges;
    return View();
}

LinearData CreateLinearGauge(DataRow row)
{
    double minimum = Convert.ToDouble(row["FreeStock"]);
    double maximum = Convert.ToDouble(row["MaxStock"]);
    double value = Convert.ToDouble(row["AvailableStock"]);
    return new LinearData() { Minimum = minimum, Maximum = maximum, Value = value };
}
```

```
}
`
```

Step 2:

Render the linear gauges based on the length of data collection that is from controller page.

```
`html
@{
    var count = 0;
    foreach (LinearData gauge in ViewBag.GaugeData)
    {
        @(Html.EJS().LinearGauge("PointerGauge" + (++count).ToString()).Border(br =>
            br.Color("black").Width(2)).Axes(axes =>
            {
                axes.LabelStyle(ls =>
                    ls.Font(font)).Minimum(gauge.Minimum).Maximum(gauge.Maximum).Ranges(range =>
                    {
                        range.Start(gauge.Minimum).End(gauge.Maximum).StartWidth(6).EndWidth(6).Color("red").Add();
                    }).Pointers(pointer =>
                    {
                        pointer.Value(gauge.Value).Add();
                    }).Add();
                }
            ).Width("550").Render())
        }
    }
`
```

Step 3:

Create a button element to call the ajax request in the button click event. In ajax call, you can update the gauge pointer values based on the ajax data.

```
`html
<button id="updatePointer" onclick="updatePointer()">Update Pointer</button>
<script>
var updatePointer = function () {
$.ajax({
    type: "POST",
```

```

url: '@Url.Action("GetServerData","maps")',
async: false,
success: function (data) {
var gauge;
for (var i = 0; i < data.length; i++) {
gauge = document.getElementById('PointerGauge' + (i+1).toString()).ej2_instances[0];
gauge.axes[0].pointers[0].value = data[i].value;
gauge.refresh();
}
}
});
}
</script>
`cs
public ActionResult GetServerData()
{
List<pointData> data1 = new List<pointData>();
data1.Add(new pointData(5600));
data1.Add(new pointData(5700));
data1.Add(new pointData(6700));
data1.Add(new pointData(6700));
return Json(data1);
}
public class pointData
{
public pointData(double value)
{
this.value = value;
}
public double value { get; set; }
}
`

```

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().LinearGauge("linear").Axes(axis=>axis.Minimum(0).Maximum(120).MajorTicks(new
{
    interval = 10,
    height = 10
}).MinorTicks(new
{
    interval = 5,
    height = 5
}).LabelStyle(new
{
    format = "c"
})).Render()
<script>
    ej.base.setCulture('de');
    ej.base.setCurrencyCode('EUR');
</script>
```

DYNAMIC.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using System.Web.Script.Serialization;
using System.Data;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
using Syncfusion.EJ2.LinearGauge;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            DataTable dt = new DataTable();
            dt.Columns.Add("Name");
            dt.Columns.Add("FullName");
            dt.Columns.Add("Stock");
            dt.Columns.Add("MaxStock");
            dt.Columns.Add("FreeStock");
            dt.Columns.Add("AvailableStock");
            dt.Columns.Add("StockCode");
            dt.Columns.Add("Status");
            dt.Rows.Add(new Object[] { "T13", "PM01 T13", 5000, 10000, 5000,
8230, "F" });
            dt.Rows.Add(new Object[] { "T14", "PM01 T14", 4000, 9000, 5000,
7230, "F" });
        }
    }
}
```

```

        dt.Rows.Add(new Object[] { "T15", "PM01 T15", 3000, 8000, 5000,
5230, "F" });
        dt.Rows.Add(new Object[] { "T16", "PM01 T16", 2000, 7000, 5000,
5230, "F" });
        List<LinearData> gauges = new List<LinearData>();
        foreach (DataRow row in dt.Rows)
            gauges.Add(CreateLinearGauge(row));
        ViewBag.GaugeData = gauges;
        return View();
    }
    [HttpPost]
    public ActionResult GetServerData()
    {
        List<pointData> data1 = new List<pointData>();
        data1.Add(new pointData(5600));
        data1.Add(new pointData(5700));
        data1.Add(new pointData(6700));
        data1.Add(new pointData(6700));
        return Json(data1);
    }
    LinearData CreateLinearGauge(DataRow row)
    {
        double minimum = Convert.ToDouble(row["FreeStock"]);
        double maximum = Convert.ToDouble(row["MaxStock"]);
        double value = Convert.ToDouble(row["AvailableStock"]);
        return new LinearData() { Minimum = minimum, Maximum = maximum,
Value = value };
    }
    public class pointData
    {
        public pointData(double value)
        {
            this.value = value;
        }
        public double value { get; set; }
    }
    public class LinearData
    {
        public double Minimum { get; set; }
        public double Maximum { get; set; }
        public double Value { get; set; }
    }
}

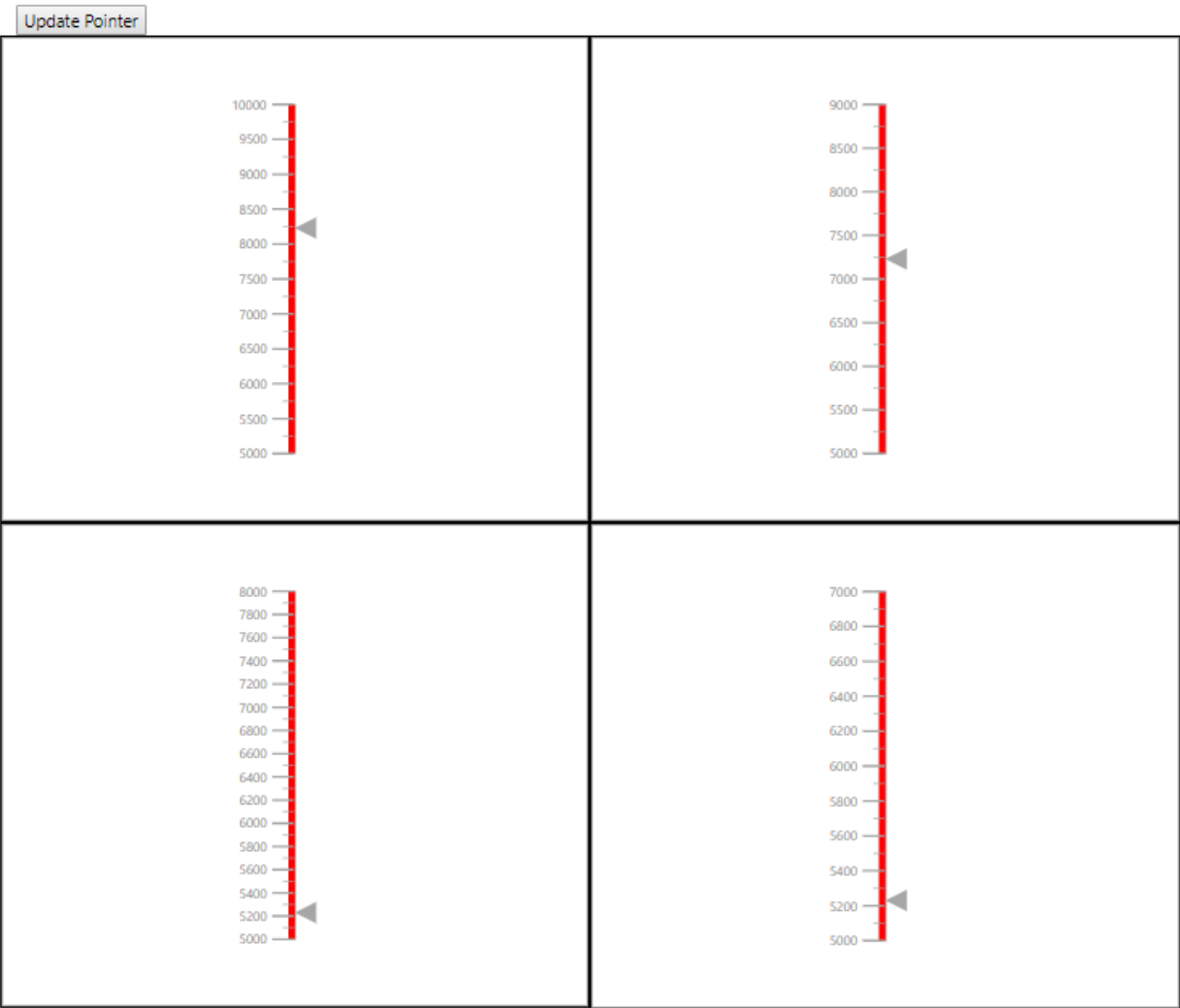
```

Sample reference

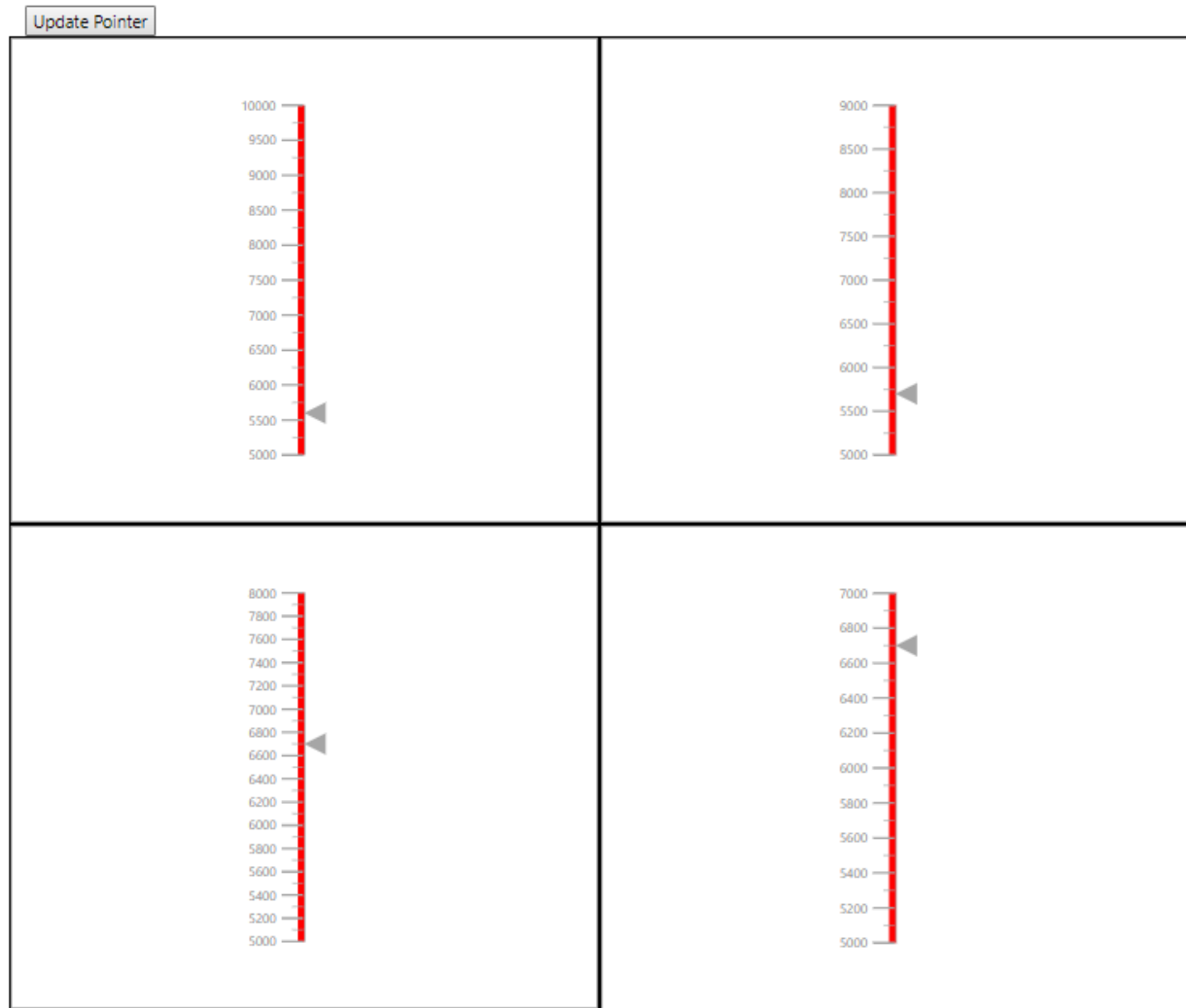
[linear Gauge sample.](#)

Screenshot

Initial Gauges



Update pointers on button click



ListBox

Getting Started with ASP.NET MVC ListBox Control

This section briefly explains about how to include [ASP.NET MVC ListBox](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
```

```
</body>
```

Add ASP.NET MVC ListBox control

Now, add the Syncfusion ASP.NET MVC ListBox control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@model List<string>
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "BadmHennessey Venominton",
    "Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
    "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
    return View(data);
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC ListBox control will be rendered in the default web browser.

```
BadmHennessey Venominton
Bugatti Chiron
Bugatti Veyron Super Sport
SSC Ultimate Aero
Koenigsegg CCR
McLaren F1
Aston Martin One- 77
Jaguar XJ220
```

Note: [View Sample in GitHub.](#)

Accessibility in List Box Component

The ListBox component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ListBox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The ListBox component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the ListBox component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the ListBox component wrapper element as **listbox**, the **UL** element as **presentation**, and its list item as **option**. |

| **aria-label** | Provides an accessible name for the ListBox component. |

| `aria-multiselectable` | Applied to the element with the ListBox role, tells assistive technologies that the list supports multiple selection. The default value is true. |

| `aria-selected` | Applied to elements with role option that are visually styled as selected to inform assistive technologies that the options are selected. |

Keyboard interaction

The ListBox component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the ListBox component.

| **Press** | **To do this** |

| --- | --- |

| Up arrow | Moves focus to the previous option. |

| Down arrow | Moves focus to the next option. |

| Home | Moves focus to first option. |

| End | Moves focus to last option. |

| Space | Changes the selection state of the focused option. |

| Ctrl + A | Selects all options in the list. |

| Ctrl + Shift + Home | Selects the focused option and all options up to the first option. |

| Ctrl + Shift + End | Selects the focused option and all options down to the last option. |

| Ctrl + (Up or Down) | Press Ctrl key with up / down arrow or mouse to select multiple items. |

Ensuring accessibility

The ListBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ListBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ListBox component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Data Binding in ASP.NET MVC List Box Control

The ListBox loads the data either from local data sources or remote data services using the [DataSource](#) property. It supports the data type of `array` or `DataManager`.

| Fields | Type | Description |

|-----|-----|-----|

| [text](#) | `string` | Specifies the display text of each list item. |

| [value](#) | `string` | Specifies the hidden data value mapped to each list item that should contain a unique value. |

| [groupBy](#) | string | Specifies the category under which the list item has to be grouped. |

| [iconCss](#) | string | Specifies the iconCss class that needs to be mapped. |

| [htmlAttributes](#) | string | Allows additional attributes to configure the elements in various ways to meet the criteria. |

Note: When binding complex data to the ListBox, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Local Data

Local data can be represented by the following ways as described below.

Array of string

The ListBox has support to load array of primitive data such as strings or numbers. Here, both value and text field acts as same.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Render()
```

DATABINDING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult databinding()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
            "Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
            "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

Array of object

The ListBox can generate its list items through an array of object data. For this, the appropriate columns should be mapped to the [Fields](#) property.

In the following example, **id** and **sports** column from complex data have been mapped to the **value** field and **text** field, respectively.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Render()
```

DATABINDING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult databinding()
        {
            List<object> Data = new List<object> {
                new { text = "Hennessey Venom", id = "list-01" },
                new { text = "Bugatti Chiron", id = "list-02" },
                new { text = "Bugatti Veyron Super Sport", id = "list-03" },
                new { text = "SSC Ultimate Aero", id = "list-04" },
                new { text = "Koenigsegg CCR", id = "list-05" },
                new { text = "McLaren F1", id = "list-06" },
                new { text = "Aston Martin One- 77", id = "list-07" },
                new { text = "Jaguar XJ220", id = "list-08" }
            };
            ViewBag.data = Data;
            return View();
        }
    }
}

```

Array of complex object

The ListBox can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [Fields](#) property.

In the following example, `Sports.Name` column from complex data have been mapped to the `text` field.

CSHTML

```

@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Fields(new Syncfusion.EJ2.DropDowns.ListBoxFieldSettings {
    Text="sports.Name"}).Render()

```

DATABINDING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult databinding()
        {
            List<object> Data = new List<object>

```

```
{
    new { id = "game0", sports = new {Name = "Badminton" } },
    new { id = "game1", sports = new {Name = "Cricket" } },
    new { id = "game2", sports = new {Name = "Football" } },
    new { id = "game3", sports = new {Name = "Golf" } },
    new { id = "game4", sports = new {Name = "Tennis" } },
    new { id = "game5", sports = new {Name = "Basket Ball" } },
    new { id = "game6", sports = new {Name = "Base Ball" } },
    new { id = "game7", sports = new {Name = "Hockey" } }
};
ViewBag.data = Data;
return View();
}
```

Remote Data

The ListBox supports retrieval of data from remote data services with the help of [DataManager](#) component. The [Query](#) property is used to fetch data from the database and bind it to the ListBox.

The following sample displays the employee names from Employee table.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.DataSource(dataManager=>

dataManager.Url("https://services.syncfusion.com/aspnet/production/api/Employees").Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.ListBoxFieldSettings { Text="FirstName",
Value="EmployeeID" }).Render()
```

DATABINDING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult databinding()
        {
            return View();
        }
    }
}
```

Drag and drop in ListBox Control

The ListBox has support to drag an item or a group of selected items and drop it within the same list box or into another list box.

The elements can be customized on drag and drop by using the following events,

| Events | Description |
|---------------------------|--|
| ----- ----- | |
| dragStart | Triggers when the selected element is being dragged. |
| drag | Triggers when the selected element is being dragged. |
| drop | Triggers when the selected element is being dropped. |

Single listbox

To drag and drop an item or group of item within the list box can be achieved by setting [allowDragAndDrop](#) property as `true`.

The following sample illustrates how to drag and drop an item within the same list box by enabling [allowDragAndDrop](#) property.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.AllowDragAndDrop(true).Render()
```

DRAGDROP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult dragdrop()
        {
            ViewBag.data = new string[] { "Australiia", "Bermuda", "Canada",
            "Cameroon", "Denmark", "France", "Finland", "Germany", "Hong kong" };
            return View();
        }
    }
}
```

Multiple listbox

In the following sample, the [allowDragAndDrop](#) property is set as `true` and [scope](#) is set as `combined-list` to enable drop and drop in both list boxes.

CSHTML

```
<div style="width:50%; margin:auto">
    <div style="float:left; width:48%">

@Html.EJS().ListBox("listbox1").DataSource((IEnumerable<object>)ViewBag.grou
pA).AllowDragAndDrop(true).Scope("combined-list").Render()
    </div>
```

```
<div style="float:right; width:48%">
@Html.EJS().ListBox("listbox2").DataSource((IEnumerable<object>)ViewBag.groupB).AllowDragAndDrop(true).Scope("combined-list").Render()
</div>
</div>
```

DRAGDROP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult dragdrop()
        {
            ViewBag.groupA = new string[] { "Austrelia", "Bermuda", "Canada", "Cameroon", "Denmark", "France", "Finland", "Germany", "Hong kong" };
            ViewBag.groupB = new string[] { "India", "Italy", "Japan", "Mexico", "Norway", "Poland", "Switzerland", "United Kingdom", "United States" };
            return View();
        }
    }
}
```

Dual list box in ASP.NET MVC ListBox Control

The dual list box allows the user to move items between two list boxes by clicking the toolbar buttons. Dual list box can be created by listing items in the [toolbarSettings](#) along with the `scope` property.

The following operations can be performed in dual list box,

| Options | Description |
|-------------|---|
| ----- ----- | |
| moveUp | Move the selected item in the upward direction within the list box. |
| moveDown | Move the selected item in the downward direction within the list box. |
| moveTo | Move the selected item to the another list box. |
| moveFrom | Move the selected item from one list box to the another list box. |
| moveAllTo | Move all the items to the another list box. |
| moveAllFrom | Move all the items from one list box to the another list box. |

The following example illustrates how to move items from `Group A` to `Group B` list box.

CSHTML

```
<div style="width:50%; margin:auto">
```

```
<div style="float:left; width:48%">

@Html.EJS().ListBox("listbox1").DataSource((IEnumerable<object>)ViewBag.groupA).Scope("#listbox2").ToolBarSettings(new
Syncfusion.EJ2.DropDowns.ListBoxToolBarSettings { Items = ViewBag.items
}).Render()
</div>
<div style="float:right; width:48%">

@Html.EJS().ListBox("listbox2").DataSource((IEnumerable<object>)ViewBag.groupB).Render()
</div>
</div>
```

DUALLISTBOX.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult duallistbox()
        {
            ViewBag.groupA = new string[] { "Australiia", "Bermuda", "Canada",
"Cameroon", "Denmark", "France", "Finland", "Germany", "Hong kong" };
            ViewBag.groupB = new string[] { "India", "Italy", "Japan",
"Mexico", "Norway", "Poland", "Switzerland", "United Kingdom", "United
States" };
            ViewBag.items = new string[] { "moveUp", "moveDown", "moveTo",
"moveFrom", "moveAllTo", "moveAllFrom" };
            return View();
        }
    }
}
```

Icons and Customization

Icons

To place the icon on a list box, set the [iconCss](#) property to **e-icons** with the required icon CSS. By default, the icon is positioned to the left side of the list.

In the following sample, icon classes are mapped with **iconCss** field.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.Data)
.Fields(new Syncfusion.EJ2.DropDowns.ListBoxFieldSettings { Text="Text",
IconCss="IconCss" }).Render()
<style>
    @@font-face {
        font-family: 'e-listbox-icons';
```

448

```

BERAOBwshQhZDBAEBBEMWQQwMDRIiNyEOExUVDgs+CwwTEhENMSkxDAgGBUAoDQkJCAcGBQYOCAo
JCgsKDASMcwwNDA0NDawXfHYUFBiQDw4MCgcGBAEBBAYHCgwODxASFBQWFhgLAAIAAAAAA78D8wA
DAOWAAAEVITUlDwcdAR8HPwcvATU/BxUjDw4VHwg/BjU/ChUPDxUfBzSBPxEVDw8VHwc7AT8RFQ8
QHwc7AT8RFSERISMPBQM/mH+/QYGBAUGBAMBAgIDBAQFBQUBBQEBaICAQIBaQIEBQoLCwwMDAs
LCwoJCQYFBQqHBAMBAQEcbAMFBQUBgUEBAMDAQMDBQMEBAUFCwsMDAwLCwsKCQkGBQUEBgUCAgE
BAwMEBAUFBgUFBAQEAgEBAgMFAwQEBQULCwwMCwwLCgoKCAcFBQQGBQICAQEDAwQEBQUGBQUEBAQ
CAQECaWUDBAQFBQsLDALDAsKCGoIBgYFBAYFAgEBAQEDAwQEBQUGBQUEBAQCAQECaWUDBAQFBQs
LDAL6/QYMCwwLCwoLA43e3ksHBgcIDhAQERERBgUFAwMCAQEBAQIDAwUFBgsYDAwLCQgGBAIBMQE
CAwQFBgcHBwYIDhAQERERBgUFAwMCAQEBAQIDAwUfKhILCgQFAwMDBAIBnAEBAQMEBQYHBwCHBw8
PEBEREQYGBAQCAgICAgIEBAYqEgSKBAQEAWMEAgGWAQEBAwQFBgcHBwCHDw8QERERBgYEBaICAgi
CAgQEBioSCwoEBAQDAwQCAZYBAQEDBAUGBwCHBwCPDxAREREGBQUEAgICAgICBAUfKhILCgQEBAM
DBAIBbgPYAQIDBAUGAAAAAAAEAAAAA/QD9AALAAATCQEXCQE3CQEnCQEMATL+zsIBMGeywv7OATL
C/s7+zgMy/s7+zsIBMv7OwgEyATLC/s4BMgAAwAAAAAD9AMyADQATABQAAABIw8aPw4zHwQVNYc
BIT8SNQkBHwEBIQMvEhIiIB4cGxgXFRQSEQ8ODAwKCQgHBgYEBwQDAQkKCw0NDw8QERIRExISExI
jIh8nK8XF/N0BTggODA0PERMVGBodIBESExMUFRb+ov6ixZwBMv2WAecBAgMEBgYHCAkKCGsLCww
MDAwNDawLDBUTEREQEA0NCgoICAYFBAMCAgEBAwMHCv6vYP7tERoRehISERAQDg4MBQUBAMDA+T
+qAFYz5wBOAAAAAABAAAAAP0A68AbgAAExcZJyEfExUPDxUzPx09AS8dIyE3JyMMztOWAWIYGBc
VFRQSEQ8ODAUFBQDAWIBAQIDBQYICgsNDhASEhUVGDU9EhMSERERE8PDw4NDQwLCwoKCQkHBwC
FBQQEAwECAgIEBAUGBwgICQoLCwwMDg0PDw8QEBEREhISExMTE/6mkgLRARXzsQEDBAYHCQoMDg8
QCQoJCgoLCwwLGRcWFBMREA4NCwoIBwQEAQGNAQIDAwUFBQCHBwkJCQsLCwwNDQ4ODxAQEBESEhI
TExQTEhISERAQEA8PDQ4NDawLCgoJCAgHBwUFBAQDAgGrAgAAABIA3gABAAAAAAAEAAAAABAAA
AAAABAA8AAQABAAAAAAACAACAEAAABAAAAAADAA8AFwABAAAAAAAEAA8AJgABAAAAAAFAAAsANQA
BAAAAAAGAA8AQABAAAAAAAKACwATwABAAAAAALABIAewADAAEECQAAAAIAjQADAAEECQABAB4
AjwADAAEECQACAA4ArQADAAEECQADAB4AUwADAAEECQAEAB4A2QADAAEECQAFABYA9wADAAEECQA
GAB4BDQADAAEECQAKAFgBKwADAAEECQALACQBgyB1LWxpc3Rib3gtaWNvbnNSZWdlbGFyZS1saXN
0Ym94LWljb25zZS1saXN0Ym94LWljb25zVmVyc2lvbiAxLjB1LWxpc3Rib3gtaWNvbnNGb250IGd
lbmVYyYXRlZCBlc2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQA
gAGUALQBsAGkAcwB0AGIAbwB4AC0AaQBjAG8AbgBzAFIAZQBnAHUAbABhAHIAZQAtAGwAaQBzAHQ
AYgBvAHgALQBpAGMAbwBuAHMAZQAtAGwAaQBzAHQAYgBvAHgALQBpAGMAbwBuAHMAVgBlAHIAcwB
pAG8AbgAgADEALgAwAGUALQBsAGkAcwB0AGIAbwB4AC0AaQBjAG8AbgBzAEYAbwBuAHQAIAbnAGU
AbgBlAHIAgYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHMAaQBvAG4AIABNAGUAdAB
yAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAA
CAAAAAAAAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAASBAgEDAQQBBQEGAQCBCAEJAQoBCwEMAAh
0b3VjaC13ZglyZXBseS1hbGwFcmVwbHkUc2F2ZS1hbGwtYXR0YWNobWVudHMkC2F2ZS1hcy13ZhN
1c2VyLXNldHRpbmdzLTAYLXdmDmFkcmVzcy1ib29rLTAzBmRlbGV0ZQdmb3J3YXJkY3VuZG9fMDE
AAAAA) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.e-list-icons {
    font-family: 'e-listbox-icons';
    speak: none;
    font-style: normal;
    font-weight: normal;
    font-variant: normal;
    text-transform: none;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
}
.e-list-touch:before {
    content: "\e700";
}
.e-list-reply-all:before {
    content: "\e701";
}
.e-list-reply:before {
    content: "\e702";
}

```

```

    }
    .e-list-save-all-attachments:before {
        content: "\e703";
    }
    .e-list-icon-save-as:before {
        content: "\e704";
    }
    .e-list-user-settings:before {
        content: "\e705";
    }
    .e-list-address-book:before {
        content: "\e706";
    }
    .e-list-delete:before {
        content: "\e707";
    }
    .e-list-forward:before {
        content: "\e708";
    }
    .e-list-undo:before {
        content: "\e709";
    }
}
</style>

```

ICONS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult icons()
        {
            List<object> Data = new List<object>{
                new { Text = "Account Settings", IconCss = "e-list-icons e-list-user-settings" },
                new { Text = "Address Book", IconCss = "e-list-icons e-list-address-book" },
                new { Text = "Forward", IconCss = "e-list-icons e-list-delete" },
                new { Text = "Reply", IconCss = "e-list-icons e-list-reply" },
                new { Text = "Reply All", IconCss = "e-list-icons e-list-reply-all" },
                new { Text = "Save All Attachments", IconCss = "e-list-icons e-list-save-all-attachments" },
                new { Text = "Save As", IconCss = "e-list-icons e-list-icon-save-as" },
                new { Text = "Touch/Mouse Mode", IconCss = "e-list-icons e-list-touch" },
                new { Text = "Undo", IconCss = "e-list-icons e-list-undo" },
            };
        }
    }
}

```

```

        ViewBag.data = Data;
        return View();
    }
}

```

Templates

ListBox items can be customized according to the requirement using [itemTemplate](#) property.

In the following sample, the items in the cart are displayed using list box template,

CSHTML

```

@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.ItemTemplate("<div class='list-wrapper'><span class='{Pic}' e-avatar e-
avatar-xlarge e-avatar-circle'></span><span class='text'>${Text}</span><span
class='description'>${Description}</span></div'>.Render()
<style>
    .e-listbox-wrapper {
        margin: auto;
        max-width: 400px;
        box-sizing: border-box;
    }
    .e-listbox-wrapper:not(.e-list-template) .e-list-item {
        padding: 0;
        position: unset;
        cursor: pointer;
        height: 76px;
        line-height: normal;
    }
    .list-wrapper {
        height: inherit;
        position: relative;
        padding: 14px 12px 14px 78px;
    }
    .list-wrapper .text,
    .list-wrapper .description {
        display: block;
        margin: 0;
        padding-bottom: 3px;
        white-space: normal;
    }
    .list-wrapper .description {
        font-size: 12px;
        font-weight: 500;
    }
    #container .e-listbox-wrapper .list-wrapper .text {
        font-weight: bold;
        font-size: 13px;
    }
    .list-wrapper .e-avatar {
        position: absolute;
        left: 5px;
        background-color: transparent;
        font-size: 22px;
        top: calc(50% - 33px);
    }

```

```

    }
    .javascript, .typeScript {
        background-image: url('images/javascript1.svg');
    }
    .angular {
        background-image: url('images/angular1.svg');
    }
    .vue {
        background-image: url('images/vue.svg');
    }
    .react {
        background-image: url('images/react.svg');
    }
}
</style>

```

TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult template()
        {
            List<object> Data = new List<object>{
                new { Text="JavaScript", Pic="javascript", Description="It is a lightweight interpreted or JIT-compiled programming language." },
                new { Text="TypeScript", Pic="typeScript", Description="It is a typed superset of Javascript that compiles to plain JavaScript." },
                new { Text="Angular", Pic="angular", Description="It is a TypeScript-based open-source web application framework." },
                new { Text="React", Pic="react", Description="A JavaScript library for building user interfaces. It can also render on the server using Node." },
                new { Text="Vue", Pic="vue", Description="A progressive framework for building user interfaces. it is incrementally adoptable." }
            };
            ViewBag.data = Data;
            return View();
        }
    }
}

```

Selection in ASP.NET MVC ListBox Control

The ListBox provides support to select an item or a group of item by mouse or keyboard action. There are two selection modes available in list box,

- Single - To select single item in the list box.
- Multiple - To select multiple items in the list box.

On selection of each list box item, [change](#) event is triggered.

Single selection

To enable single selection in the list box, [mode](#) should be set as **Single** in [SelectionSettings](#) property.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.SelectionSettings(new Syncfusion.EJ2.DropDowns.ListBoxSelectionSettings {
Mode = Syncfusion.EJ2.DropDowns.SelectionMode.Single}).Render()
```

SELECTION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult selection()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
"Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
"Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

Multiple selection

To enable multiple selection in the list box, [mode](#) should be set as **Multiple** in [selectionSettings](#) property.

To select multiple items, use the SHIFT, CTRL, and arrow keys to make selections.

Note: By default, the selection mode is set as **Multiple**.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.SelectionSettings(new Syncfusion.EJ2.DropDowns.ListBoxSelectionSettings {
Mode = Syncfusion.EJ2.DropDowns.SelectionMode.Multiple}).Render()
```

SELECTION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult selection()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
"Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
"Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

```
{
    public class ListBoxController : Controller
    {
        public IActionResult selection()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
            "Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
            "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

Note: To select all the items in the list box, [ShowSelectAll](#) method can also be used.

sorting and grouping

Sorting

The ListBox supports sorting of available items in the alphabetical order that can be either ascending or descending. This can be achieved using [sortOrder](#) property. Sort order can be **None**, **Ascending** or **Descending**.

In the following example, the **SortOrder** is set as **Descending**.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.SortOrder(ViewBag.value).Render()
```

SORTING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult sorting()
        {
            ViewBag.data = new string[] { "Austrelia", "Bermuda", "Canada",
            "Cameroon", "Denmark", "France", "Finland", "Germany", "Hong kong" };
            ViewBag.value = "Descending";
            return View();
        }
    }
}
```

Grouping

The ListBox supports to wrap the nested element into a group based on its category. The category of each list item can be mapped with [groupBy](#) field in the data table.

In the following example, vegetables are grouped based on its category.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Fields(new Syncfusion.EJ2.DropDowns.ListBoxFieldSettings {
    GroupBy="Category", Text="Vegetable", Value="Id"}).Render()
```

GROUPING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult grouping()
        {
            List<object> Data = new List<object> {
                new { Vegetable = "Cabbage", Category = "Leafy and Salad", Id =
"item1" },
                new { Vegetable = "Spinach", Category = "Leafy and Salad", Id =
"item2" },
                new { Vegetable = "Wheat grass", Category = "Leafy and Salad",
Id = "item3" },
                new { Vegetable = "Yarrow", Category = "Leafy and Salad", Id =
"item4" },
                new { Vegetable = "Pumpkins", Category = "Leafy and Salad", Id =
"item5" },
                new { Vegetable = "Chickpea", Category = "Beans", Id = "item6"
},
                new { Vegetable = "Green bean", Category = "Beans", Id = "item7"
},
                new { Vegetable = "Horse gram", Category = "Beans", Id = "item8"
},
                new { Vegetable = "Garlic", Category = "Bulb and Stem", Id =
"item9" },
                new { Vegetable = "Nopal", Category = "Bulb and Stem", Id =
"item10" },
                new { Vegetable = "Onion", Category = "Bulb and Stem", Id =
"item11" }
            };
            ViewBag.data = Data;
            return View();
        }
    }
}
```

Styles and Appearances

To modify the ListBox appearance, you need to override the default CSS of ListBox component. Find the list of CSS classes and its corresponding section in ListBox component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

- |.e-listbox-wrapper|To customize the listbox wrapper
- |.e-list-parent .e-list-item|To customize the listbox list items
- |.e-list-parent .e-list-item:hover|To customize the listbox list items on hover
- |.e-list-parent .e-list-item.e-selected|To customize the listbox selected list item
- |.e-listboxtool-wrapper .e-listbox-tool|To customize the listbox toolbar
- |.e-listboxtool-wrapper .e-listbox-tool .e-btn|To customize the listbox toolbar button
- |.e-listboxtool-wrapper .e-listbox-tool .e-btn .e-btn-icon.e-icons::before|To customize the listbox toolbar icon

Horizontal ListBox

You can use [CssClass](#) property to display the Listbox horizontally.

CSHTML

```
<div id="listbox-container">
@Html.EJS().ListBox("listbox").CssClass("e-horizontal-
listbox").DataSource((IEnumerable<object>)ViewBag.data).Render()
</div>
<style>
#listbox-container {
    margin: 20px auto 0;
    width: 250px;
}
/* Custom css for horizontal listbox */
.e-horizontal-listbox .e-list-parent {
    display: inline-flex;
    align-items: center;
}
.e-horizontal-listbox {
    overflow-y: hidden;
    height: 100px;
}
.e-horizontal-listbox .e-list-parent .e-list-item {
    width: max-content;
    line-height: 100px;
    height: 100px;
}
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
            "Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
            "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

How To

Add items to the list box

To add an item or multiple items, [addItem](#) method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items will be added while clicking Add Items button.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Render()
<button class="e-btn" id="additem">Add Items</button>
<script>
    document.getElementById("additem").onclick = function () {
        var items = ['Bugatti Veyron Super Sport', 'SSC Ultimate Aero'];
        var listboxobj =
document.getElementById("listbox").ej2_instances[0];
        if (!listboxobj.getDataByValue('Bugatti Veyron Super Sport')) {
            listboxobj.addItem(items);
        }
    }
}
```

ADDITEM.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult additem()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
            "Bugatti Chiron", "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77",
            "Jaguar XJ220" };
            return View();
        }
    }
}
```

```
}
}
```

Enable or disable items

To enable or disable items in the list box, [enableItems](#) method can be used. In the following example, the Bugatti Veyron Super Sport and SSC Ultimate Aero items are disabled by default and by clicking Enable Items buttons, the disabled items will be enabled.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Created("create").Render()
<button class="e-btn" id="enableitem">Enable Items</button>
<script>
    var disableItems = ['Bugatti Veyron Super Sport', 'SSC Ultimate Aero'];
    function create() {
        var listboxobj =
document.getElementById("listbox").ej2_instances[0];
        listboxobj.enableItems(disableItems, false);
    }
    document.getElementById('enableitem').onclick = function () {
        var listboxobj1 =
document.getElementById("listbox").ej2_instances[0];
        listboxobj1.enableItems(disableItems, true);
    }
</script>
```

ENABLEITEM.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult enableitem()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
"Bugatti Veyron Super Sport", "SSC Ultimate Aero", "Bugatti Chiron",
"Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

Enable Scroller

The ListBox supports scrolling and it can be achieved by restricting the height of the list box using [height](#) property.

In the following sample, [height](#) of the list box is restricted to 250px.

CSHTML

```
@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Height("250px").Render()
```

SCROLLER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult scroller()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
"Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
"Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}
```

Form submit to the list box

In the following code snippet, the value that is in selected state will be sent on form submit.

CSHTML

```
<form>

@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Render()
    @Html.EJS().Button("button").IsPrimary(true).Content("Submit").Render()
</form>
```

FORMSUBMIT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult formsubmit()
        {

```

```

        ViewBag.data = new string[] { "BadmHennessey Venominton",
        "Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
        "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
        return View();
    }
}

```

Select items to the list box

In the following example, **Bugatti Chiron** is selected using [selectItems](#) method.

CSHTML

```

@Html.EJS().ListBox("listbox").DataSource((IEnumerable<object>)ViewBag.data)
.Created("create").Render()
<script>
    function create() {
        var listboxobj =
document.getElementById("listbox").ej2_instances[0];
        listboxobj.selectItems(['Bugatti Chiron'])
    }
</script>

```

SELECTITEM.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class ListBoxController : Controller
    {
        public IActionResult selectitem()
        {
            ViewBag.data = new string[] { "BadmHennessey Venominton",
            "Bugatti Chiron", "Bugatti Veyron Super Sport", "SSC Ultimate Aero",
            "Koenigsegg CCR", "McLaren F1", "Aston Martin One- 77", "Jaguar XJ220" };
            return View();
        }
    }
}

```

ListView

Getting Started with ASP.NET MVC ListView control

This section briefly explains about how to include [ASP.NET MVC ListView](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC

controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC ListView control

Now, add the Syncfusion ASP.NET MVC ListView control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().ListView("listview").Render()
```

Bind dataSource

After initialization, populate the ListView with data using the [DataSource](#) property. Here, the JSON values are passed to the ListView control.

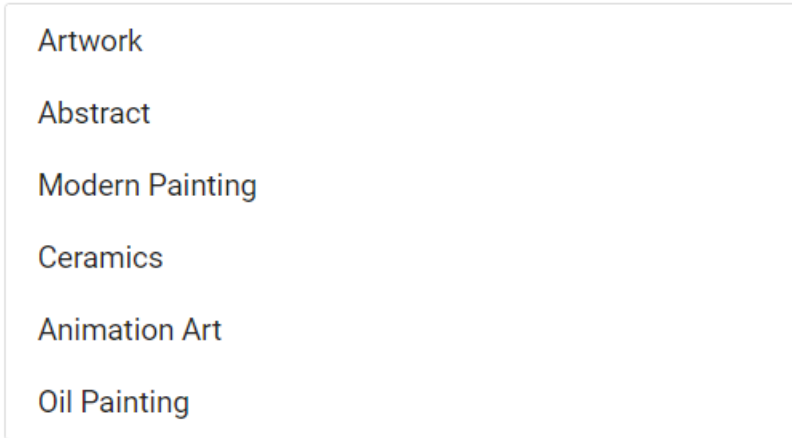
CSHTML

```
@model List<object>
@Html.EJS().ListView("listview").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public IActionResult Index()
{
    //define the array of JSON
    List<object> data = new List<object>();
    data.Add(new { text= "Artwork", id= "01" });
    data.Add(new { text= "Abstract", id= "02" });
    data.Add(new { text= "Modern Painting", id= "03" });
    data.Add(new { text= "Ceramics", id= "04" });
    data.Add(new { text= "Animation Art", id= "05" });
    data.Add(new { text= "Oil Painting", id= "06" });
    return View(data);
}
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC ListView control will be rendered in the default web browser.



Note: [View Sample in GitHub.](#)

See also

- [Data Binding Types](#)
- [ListView customization](#)
- [Virtualization](#)

Note: You can refer to our [ASP.NET MVC ListView](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC ListView Example](#) that shows you how to render and configure the ListView in ASP.NET MVC.

Data binding

ListView provides an option to load the data either from local dataSource or remote data services. This can be done through the dataSource property that supports the data type of array or [DataManager](#).

ListView supports different kind of data services such as OData, OData V4, and Web API, and data formats like XML, JSON, and, JSONP with the help of DataManager Adaptors.

| Fields | Type | Description |
|-----------|---------|---|
| id | string | Specifies ID attribute of list item, mapped in dataSource. |
| text | string | Specifies list item display text field. |
| isChecked | boolean | Specifies checked status of list item. |
| isVisible | boolean | Specifies visibility state of list item. |
| enabled | boolean | Specifies enabled state of list item. |
| iconCss | string | Specifies the icon class of each list item that will be added before to the list item text. |
| child | string | Specifies child dataSource fields. |
| tooltip | string | Specifies tooltip title text field. |
| groupBy | string | Specifies category of each list item. |

| sortBy | string | Specifies sorting field, that is used to sort the listview data. |

| htmlAttributes | string | Specifies list item html attributes field. |

Note: When complex data bind to ListView, you should map the fields properly. Otherwise, the ListView properties remain as undefined or null.

Bind to local data

Local data can be represented in two ways, they are as follows:

- Array of simple data.
- Array of JSON data.

Array of simple data

ListView supports to load the array of primitive data like string and numbers. Here, both value and text field act as same.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).Render()
<style>
    #element {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
        ViewBag.dataSource = new string[] { "Artwork", "Abstract", "Modern
Painting", "Ceramics", "Animation Art", "Oil Painting" };
        return View();
    }
}
```

Array of JSON data

ListView can generate its list items through an array of complex data. To get it work properly, you should map the appropriate columns to the field property.

In the following example, role column is mapped with the text field.

CSHTML

```
@using Syncfusion.EJ2
```

```
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).Fields(new ListViewFieldSettings { Text = "Name", Tooltip= "Name" ,
Id="id"}).ShowHeader(true).HeaderTitle("Device settings").Render()
<style>
    #element {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listdata = new List<object>();
        listdata.Add(new
        {
            Name = "Display",
            id = "list-01"
        }); listdata.Add(new
        {
            Name = "Notification",
            id = "list-02"
        }); listdata.Add(new
        {
            Name = "Sound",
            id = "list-03"
        }); listdata.Add(new
        {
            Name = "Apps",
            id = "list-04"
        }); listdata.Add(new
        {
            Name = "Storage",
            id = "list-05"
        }); listdata.Add(new
        {
            Name = "Battery",
            id = "list-06"
        });
        ViewBag.dataSource = listdata;
        return View();
    }
}
```

Bind to remote data

The ListView supports to retrieve the data from remote data services with the help of DataManager component. The Query property allows to fetch data and return it to the ListView from the database.

In the following sample, first 6 products from the Product table of NorthWind data service are displayed.

CSHTML

```
@using Syncfusion.EJ2
<div class="control-section">
    @Html.EJS().ListView("remotelist").Enable(true).DataSource(dataManger =>
    {
        dataManger.Url("//js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/").CrossDomain(true);
    }).Query("new
ej.data.Query().from('Products').select('ProductID,ProductName').take(10)").Fields(new Syncfusion.EJ2.Lists.ListViewFieldSettings { Id = "ProductID" ,
Text = "ProductName" }).ShowHeader(true).HeaderTitle("Products").Render()
</div>
```

LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
        return View();
    }
}
```

Output be like the below.

| Products |
|------------------------------|
| Chai |
| Chang |
| Aniseed Syrup |
| Chef Anton's Cajun Seasoning |
| Chef Anton's Gumbo Mix |
| Grandma's Boysenberry Spread |

Grouping

The ListView supports to wrap the nested element into a group based on the category. The category of each list item can be mapped with `groupBy` field in the data table, that also supports single-level navigation.

In the following sample, The cars are grouped based on its category by using the `groupBy` field.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { GroupBy = "category" }).Render()
<style>
    #element {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>

```

LISTCONTROLLER.CS

```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listdata = new List<object>();
        listdata.Add(new
        {
            text = "Audi A4",
            id = "9bdb",
            category = "Audi"
        }); listdata.Add(new
        {
            text = "Audi A5",
            id = "4589",
            category = "Audi"
        }); listdata.Add(new
        {
            text = "BMW 501",
            id = "f849",
            category = "BMW"
        }); listdata.Add(new
        {
            text = "BMW 502",
            id = "7aff",
            category = "BMW"
        });
        ViewBag.dataSource = listdata;
        return View();
    }
}

```

Output be like the below.

| |
|-------------|
| Audi |
| Audi A4 |
| Audi A5 |
| BMW |
| BMW 501 |
| BMW 502 |

Customization

The grouping header can be customized by using the `groupTemplate` property for both inline and fixed group header. The complete customization description and explanation with an example is given in the following link.

[ASP .NET Core ListView - Group Template.](#)

Checklist in ASP.NET MVC Listview Control

The ListView supports checkbox in default and group-lists which is used to select multiple items. The checkbox can be enabled by the `showCheckBox` property.

The Checkbox will be useful in the scenario where we need to select multiple options. For Example, in Shipping cart we can be able to select or unselect the desired items before checkout and also it will be useful in selecting multiple items that belongs to same category using the group list.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("element").DataSource((IEnumerable<object>) ViewBag.data
Source).Fields(new ListViewFieldSettings { Text = "text",
Id="id"}).ShowCheckBox(true).Render()
<style>
    #element {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

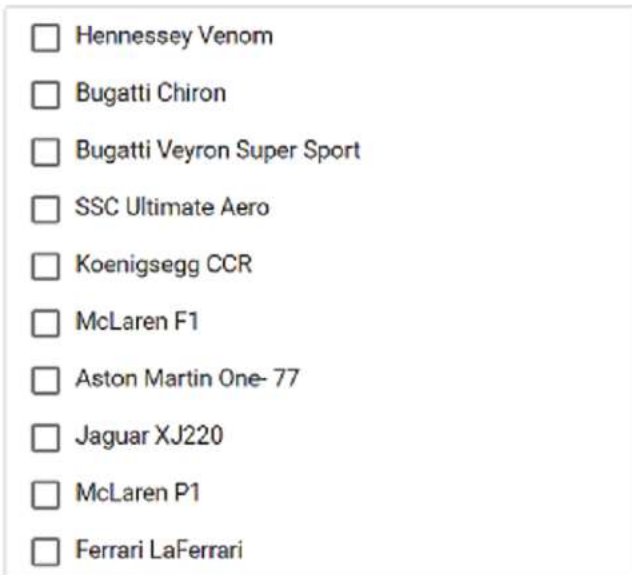
LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
```



```
List<object> listdata = new List<object>();
listdata.Add(new
{
    text = "Hennessey Venom",
    id = "list-01"
}); listdata.Add(new
{
    text = "Bugatti Chiron",
    id = "list-02"
}); listdata.Add(new
{
    text = "Bugatti Veyron Super Sport",
    id = "list-03"
}); listdata.Add(new
{
    text = "SSC Ultimate Aero",
    id = "list-04"
}); listdata.Add(new
{
    text = "Koenigsegg CCR",
    id = "list-05"
}); listdata.Add(new
{
    text = "McLaren F1",
    id = "list-06"
}); listdata.Add(new
{
    text = "Aston Martin One- 77",
    id = "list-07"
}); listdata.Add(new
{
    text = "Jaguar XJ220",
    id = "list-08"
}); listdata.Add(new
{
    text = "McLaren P1",
    id = "list-09"
}); listdata.Add(new
{
    text = "Ferrari LaFerrari",
    id = "list-10"
});
ViewBag.dataSource = listdata;
return View();
}
```

Output be like the below.



Checkbox Position

In ListView the checkbox can be positioned into either **Left** or **Right** side of the list-item text. This can be achieved by **checkboxPosition** property. By default, checkbox will be positioned to **Left** of list-item text.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).Fields(new ListViewFieldSettings { Text = "text",
Id="id"}).ShowCheckBox(true).CheckBoxPosition(CheckBoxPosition.Right).Render
()
<style>
    #element {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listdata = new List<object>();
        listdata.Add(new
        {
            text = "Hennessey Venom",
            id = "list-01"
        }); listdata.Add(new
        {
```

```
        text = "Bugatti Chiron",
        id = "list-02"
    }); listdata.Add(new
    {
        text = "Bugatti Veyron Super Sport",
        id = "list-03"
    }); listdata.Add(new
    {
        text = "SSC Ultimate Aero",
        id = "list-04"
    }); listdata.Add(new
    {
        text = "Koenigsegg CCR",
        id = "list-05"
    }); listdata.Add(new
    {
        text = "McLaren F1",
        id = "list-06"
    }); listdata.Add(new
    {
        text = "Aston Martin One- 77",
        id = "list-07"
    }); listdata.Add(new
    {
        text = "Jaguar XJ220",
        id = "list-08"
    }); listdata.Add(new
    {
        text = "McLaren P1",
        id = "list-09"
    }); listdata.Add(new
    {
        text = "Ferrari LaFerrari",
        id = "list-10"
    });
    ViewBag.dataSource = listdata;
    return View();
}
}
```

Output be like the below.

| | |
|----------------------------|--------------------------|
| Hennessey Venom | <input type="checkbox"/> |
| Bugatti Chiron | <input type="checkbox"/> |
| Bugatti Veyron Super Sport | <input type="checkbox"/> |
| SSC Ultimate Aero | <input type="checkbox"/> |
| Koenigsegg CCR | <input type="checkbox"/> |
| McLaren F1 | <input type="checkbox"/> |
| Aston Martin One- 77 | <input type="checkbox"/> |
| Jaguar XJ220 | <input type="checkbox"/> |
| McLaren P1 | <input type="checkbox"/> |
| Ferrari LaFerrari | <input type="checkbox"/> |

Nested List

The ListView component supports Nested list. For that, the child property should be defined for the nested list in the array of JSON.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("listview").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { Tooltip = "text"
}).ShowHeader(true).HeaderTitle("Continent").Render()
<style>
    #listview {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
```

LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listdata = new List<object>();
        listdata.Add(new
        {
            text = "Asia",
            id = "01",
            category = "Continent",
            child = new List<object>() { new { text = "India", id = "1",
category = "Asia",
```

```

        child= new List<object>() {
            new { id= "1001", text= "Delhi", category= "India"
        },
            new {text= "Kashmir", id= "1002", category=
"India"},
            new { text= "Goa",id= "1003", category= "India" }
        },
        new { text = "China",id = "2",category = "Asia",
        child = new List<object>() {
            new { text = "Zhejiang", id = "2001", category =
"China" },
            new {text= "Hunan",id= "2002", category= "China"},
            new { text= "Shandong", id= "2003",category=
"China"}
        }
    }
});
listdata.Add(new
{
    text = "North America",
    id = "02",
    category = "Continent",
    child = new List<object>() { new { text = "USA", id = "3",
category = "North America",
        child= new List<object>() {
            new {text= "California", id= "3001", category=
"USA"},
            new { text= "New York",id= "3002", category= "USA"
        },
            new { text= "Florida",id= "3003", category= "USA" }
        },
        new { text = "Canada",id = "4",category = "North America",
        child = new List<object>() {
            new { text = "Ontario", id = "4001", category =
"Canada" },
            new {text= "Alberta",id= "4002", category=
"Canada"},
            new { text= "Manitoba", id= "4003",category=
"Canada"}
        }
    }
});
listdata.Add(new
{
    text = "Europe",
    id = "03",
    category = "Continent",
    child = new List<object>() { new { text = "Germany", id = "5",
category = "Europe",
        child= new List<object>() {
            new {text= "Berlin", id= "5001", category=
"Germany"},

```

```

        new { text= "Bavaria",id= "5002", category=
"Germany" },
        new { text= "Hesse",id= "5003", category= "Germany"
    }
    },
    new { text = "France",id = "6",category = "Europe",
        child = new List<object>() {
            new { text = "Paris", id = "6001", category =
"France" },
            new {text= "Lyon",id= "6002", category= "France"},
            new { text= "Marseille", id= "6003",category=
"France"}
        }
    }
});
listdata.Add(new
{
    text = "Australia",
    id = "04",
    category = "Continent",
    child = new List<object>() { new { text = "Australia", id = "7",
category = "Australia",
        child= new List<object>() {
            new {text= "Sydney", id= "7001", category=
"Australia"},
            new { text= "Melbourne",id= "7002", category=
"Australia" },
            new { text= "Brisbane",id= "7003", category=
"Australia" }
        },
        new { text = "New Zealand",id = "8",category = "Australia",
            child = new List<object>() {
                new { text = "Milford Sound", id = "8001", category
= "New Zealand" },
                new {text= "Tongariro National Park",id= "8002",
category= "New Zealand"},
                new { text= "Fiordland National Park", id=
"8003",category= "New Zealand"}
            }
        },
    });
listdata.Add(new
{
    text = "Africa",
    id = "05",
    category = "Continent",
    child = new List<object>() { new { text = "Morocco", id = "9",
category = "Africa",
        child= new List<object>() {
            new {text= "Rabat", id= "9001", category=
"Morocco"},
            new { text= "Toubkal",id= "9002", category=
"Morocco" },

```

```

        new { text= "Todgha Gorge",id= "9003", category=
"Morocco" }
    },
    new { text = "South Africa",id = "10",category = "Africa",
    child = new List<object>() {
        new { text = "Cape Town", id = "10001", category =
"South Africa" },
        new {text= "Pretoria",id= "10002", category=
"South Africa"},
        new { text= "Bloemfontein", id= "10003",category=
"South Africa"}
    }
    },
    });
ViewBag.dataSource = listdata;
return View();
}
}

```

Output be like the below.

| Continent | |
|---------------|---|
| Asia | > |
| North America | > |
| Europe | > |
| Australia | > |
| Africa | > |

Customizing templates

The ListView component is designed to customize each list items and group title. It uses Essential JS2 **Template engine** to render the elements.

Header Template

ListView header can be customized with the help of the **headerTemplate** property.

To customize header template in your application, set your customized template string to **headerTemplate** property along with **showHeader** property as **true** to display the ListView header.

In the following example, we have rendered Listview with customized header which contains search, add and sort buttons.

CSHTML

```
@using Syncfusion.EJ2
```

```

@using Syncfusion.EJ2.Lists
@{ var headerTemplate = "<div class='headerContainer'><span
class='fruitHeader'>Fruits</span>" +
    "<button id='search'></button><button
id='add'></button><button id='sort'></button></div>";
}

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.fruitsdata).ShowHeader(true).HeaderTemplate(headerTemplate).ActionComplete("renderHeaderButtons").Render()
<style>
    #element {
        display: block;
        max-width: 353px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #element .e-list-header {
        padding: 0;
    }
    #element .headerContainer {
        width: 350px;
        height: 48px;
        line-height: 48px;
        background: rgb(2, 120, 215);
        color: white;
        margin-bottom: 3px;
    }
    .headerContainer .fruitHeader {
        margin-left: 20px;
        font-weight: 500;
        font-size: 22px;
    }
    .headerContainer #add,
    .headerContainer #sort,
    .headerContainer #search {
        float: right;
        margin-right: 15px;
        margin-top: 7px;
        background: white;
        color: black
    }
    .headerContainer .e-search-icon::before {
        content: '\e961';
        color: lightslategray;
    }
    .headerContainer .e-add-icon::before {
        content: '\e823';
    }
    .headerContainer .e-sort-icon::before {
        content: '\e840';
    }
</style>
<script>
    function renderHeaderButtons() {
        ['search', 'sort', 'add'].forEach((item) => {

```



```

        new ej.buttons.Button({ iconCss: `e-icons e-${item}-icon`,
cssClass: 'e-small e-round', isPrimary: true }, `#${item}`)
    });
}
</script>

```

LISTCONTROLLER.CS

```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        //define the array of JSON
        List<object> fruitsdata = new List<object>();
        fruitsdata.Add(new { text = "Date", id = "1", imgUrl = "../dates.jpg"
    });
        fruitsdata.Add(new { text = "Fig", id = "2", imgUrl = "../fig.jpg"
    });
        fruitsdata.Add(new { text = "Apple", id = "3", imgUrl =
        "../apple.png" });
        fruitsdata.Add(new { text = "Apricot", id = "4", imgUrl =
        "../apricot.jpg" });
        fruitsdata.Add(new { text = "Grape", id = "5", imgUrl =
        "../grape.jpg" });
        fruitsdata.Add(new { text = "Strawberry", id = "6", imgUrl =
        "../strawberry.jpg" });
        fruitsdata.Add(new { text = "Pineapple", id = "7", imgUrl =
        "../pineapple.jpg" });
        fruitsdata.Add(new { text = "Melon", id = "8", imgUrl =
        "../melon.jpg" });
        fruitsdata.Add(new { text = "Lemon", id = "9", imgUrl =
        "../lemon.jpg" });
        fruitsdata.Add(new { text = "Cherry", id = "10", imgUrl =
        "../cherry.jpg" });
        ViewBag.fruitsdata = fruitsdata;
        return View();
    }
}

```

Template

ListView items can be customized with the help of the `template` property.

To customize list items in your application, set your customized template string to `template` property.

We provided the following built-in CSS classes to customize the list-items. Refer to the following table.

| CSS class | Description |
|-----------|-------------|
| ----- | ----- |

| e-list-template, e-list-wrapper | These classes are used to differentiate normal and template rendering, which are mandatory for template rendering. The `e-list-template` class should be added to the root of the ListView element and `e-list-wrapper` class should be added to the template element wrapper.

| e-list-content | This class is used to align list content and it should be added to the content element
`

 <div class="e-list-wrapper">
ListItem
</div>|`

| e-list-avatar | This class is used for avatar customization. It should be added to the template element wrapper. After adding it, we can customize our element with [Avatar](#) classes
`

 <div class="e-list-wrappere-list-avatar">
 MR
ListItem
</div>|`

| e-list-avatar-right | This class is used to align avatar to right side of the list item. It should be added to the template element wrapper. After adding it, we can customize our element with [Avatar](#) classes
`

 <div class="e-list-wrappere-list-avatar-right">
 ListItem
MR
</div>|`

| e-list-badge | This class is used for badge customization .It should be added to the template element wrapper. After adding it, we can customize our element with [Badge](#) classes
`

 <div class="e-list-wrappere-list-badge">
 ListItem
MR
</div>|`

| e-list-multi-line | This class is used for multi-line customization. It should be added to the template element wrapper. After adding it, we can customize List item's header and description
`

<div class="e-list-wrappere-list-multi-line">
 ListItem
</div>|`

| e-list-item-header | This class is used to align a list header and it should be added to the header element along with the multi-line class
`

 <div class="e-list-wrappere-list-multi-line">
 ListItem Header
 ListItem
</div>|`

In the following example, we have customized list items with built-in CSS classes.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@{ var template = "<div class='e-list-wrapper e-list-multi-line e-list-avatar'>" +
    "${if (avatar!='')}" +
    "<span class='e-avatar e-avatar-circle'>${avatar}</span>" +
    "${else}" +
    "<span class='${pic} e-avatar e-avatar-circle'> </span>" +
    "${/if}" +
    "<span class='e-list-item-header'>${text}</span>" +
    "<span class='e-list-content'>${contact}</span>" +
    "</div>";
}

@Html.EJS().ListView("List").DataSource((IEnumerable<object>) ViewBag.listData).Template(template).CssClass("e-list-template").SortOrder(SortOrder.Ascending).Width(350).HeaderTitle("Contacts")
```

```

.ShowHeader(true).Fields(new ListViewFieldSettings { Text =
"name"}).Render()
<style>
    #List {
        margin: 0 auto;
        font-size: 15px;
        border: 1px solid #ccc;
    }
    .ic01 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/1.png");
    }
    .pic02 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/3.png");
    }
    .pic03 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/5.png");
    }
    .pic04 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/2.png");
    }
    #List .e-list-item:nth-child(1) .e-avatar {
        background-color: #039be5;
    }
    #List .e-list-item:nth-child(2) .e-avatar {
        background-color: #e91e63;
    }
    #List .e-list-item:nth-child(6) .e-avatar {
        background-color: #009688;
    }
    #List .e-list-item:nth-child(8) .e-avatar {
        background-color: #000088;
    }
</style>

```

LISTCONTROLLER.CS

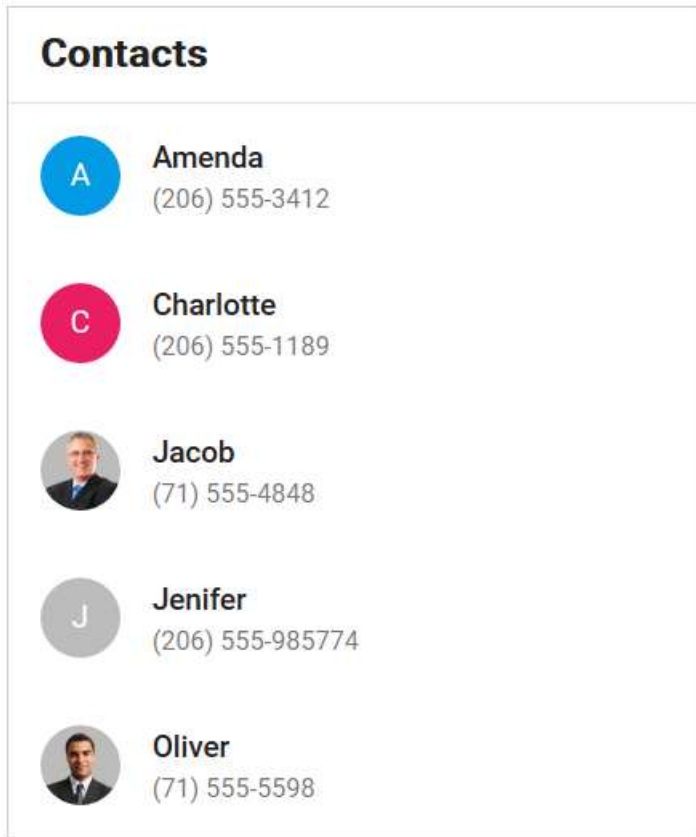
```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listData = new List<object>();
        listData.Add(new
        {
            text = "Jenifer",
            contact = "(206) 555-985774",
            id = "1",
            avatar = "J",
            pic = "pic01"
        });
        listData.Add(new
        {
            text = "Amenda",

```

```
        contact = "(206) 555-3412",
        id = "2",
        avatar = "A",
        pic = ""
    });
    listData.Add(new
    {
        text = "Isabella",
        contact = "(206) 555-8122",
        id = "4",
        avatar = "",
        pic = "pic02"
    });
    listData.Add(new
    {
        text = "William ",
        contact = "(206) 555-9482",
        id = "5",
        avatar = "W",
        pic = ""
    });
    listData.Add(new
    {
        text = "Jacob",
        contact = "(71) 555-4848",
        id = "6",
        avatar = "",
        pic = "pic04"
    });
    listData.Add(new
    {
        text = "Matthew",
        contact = "(71) 555-7773",
        id = "7",
        avatar = "M",
        pic = ""
    });
    listData.Add(new
    {
        text = "Oliver",
        contact = "(71) 555-5598",
        id = "8",
        avatar = "",
        pic = "pic03"
    });
    listData.Add(new
    {
        text = "Charlotte",
        contact = "(206) 555-1189",
        id = "9",
        avatar = "C",
        pic = ""
    });
    ViewBag.listData = listData;
    return View();
}
```

Output be like the below.



Group template

ListView group header can be customized with the help of the `groupTemplate` property.

To customize the group template in your application, set your customized template string to `groupTemplate` property.

In the following example, we have grouped Listview based on the category. The category of each list item should be mapped with `groupBy` field of the data. We have also displayed grouped list items count in the group list header.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@{ var template = "<div class='e-list-wrapper e-list-multi-line e-list-
avatar'>" +
    "<img class='e-avatar e-avatar-circle' src=${image}
style='background:#BCBCBC' />" +
    "<span class='e-list-item-header'>${Name}</span>" +
    "<span class='e-list-content'>${contact}</span>" +
    "</div>";
    var groupTemplate = "<div><span
class='category'>${items[0].category}</span> <span class='count'>
${items.length} Item(s)</span></div>";
```

```

    }

@Html.EJS().ListView("List").DataSource((IEnumerable<object>)ViewBag.listData).Template(template).GroupTemplate(groupTemplate).CssClass("e-list-template").Width(350).Fields(new ListViewFieldSettings { Text = "name", GroupBy = "category" }).Render()
<style>
    #List {
        display: block;
        margin: auto;
        font-size: 15px;
        border: 1px solid;
        border-color: #ccc;
        border-color: #00001f;
        width: 350px;
    }

    #List .e-list-group-item {
        height: 56px;
        line-height: 56px;
    }

    #List .count {
        float: right;
    }
</style>

```

LISTCONTROLLER.CS

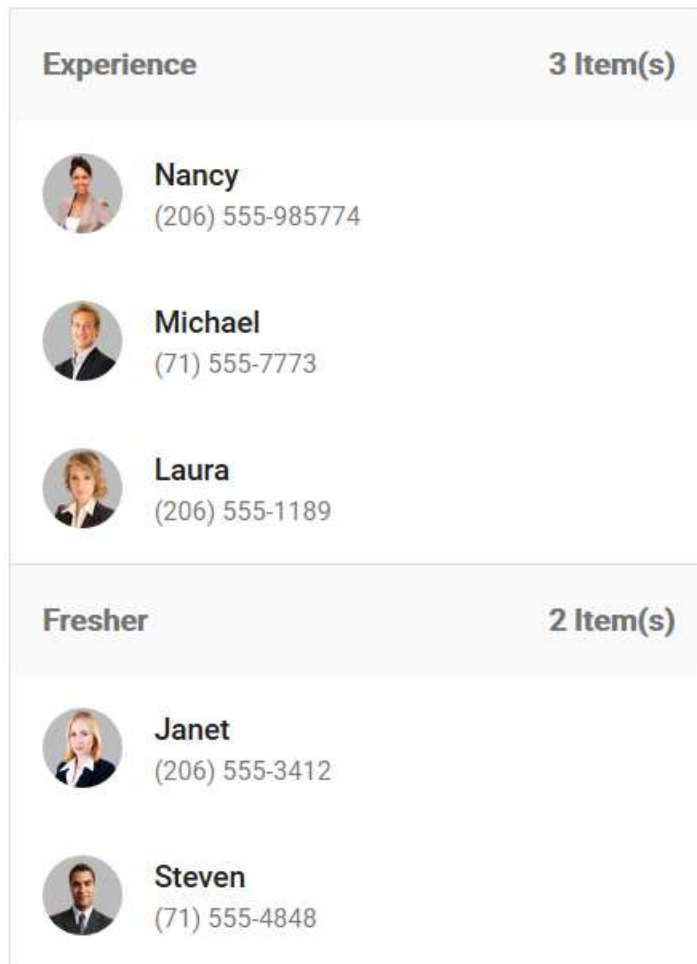
```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        //define the array of JSON
        List<object> listData = new List<object>();
        listData.Add(new { Name = "Nancy", contact = "(206) 555-985774", id = "1", image = "https://ej2.syncfusion.com/demos/src/grid/images/1.png", category = "Experience" });
        listData.Add(new { Name = "Janet", contact = "(206) 555-3412", id = "2", image = "https://ej2.syncfusion.com/demos/src/grid/images/3.png", category = "Fresher" });
        listData.Add(new { Name = "Margaret", contact = "(206) 555-8122", id = "4", image = "https://ej2.syncfusion.com/demos/src/grid/images/4.png", category = "Experience" });
        listData.Add(new { Name = "Andrew ", contact = "(206) 555-9482", id = "5", image = "https://ej2.syncfusion.com/demos/src/grid/images/2.png", category = "Experience" });
        listData.Add(new { Name = "Steven", contact = "(71) 555-4848", id = "6", image = "https://ej2.syncfusion.com/demos/src/grid/images/5.png", category = "Fresher" });
        listData.Add(new { Name = "Michael", contact = "(71) 555-7773", id = "7", image = "https://ej2.syncfusion.com/demos/src/grid/images/6.png", category = "Experience" });
        listData.Add(new { Name = "Robert", contact = "(71) 555-5598", id = "8", image = "https://ej2.syncfusion.com/demos/src/grid/images/7.png", category = "Fresher" });
    }
}

```

```
listData.Add(new { Name = "Laura", contact = "(206) 555-1189", id =  
"9", image = "https://ej2.syncfusion.com/demos/src/grid/images/8.png",  
category = "Experience" });  
ViewBag.listData = listData;  
return View();  
}  
}
```

Output be like the below.



UI Virtualization

UI virtualization loads only viewable list items in a view port, which will improve the ListView performance while loading a large number of data.

Getting started

UI virtualization can be enabled in the ListView by setting the [enableVirtualization](#) property to true.

It has two types of scrollers as follows:

Window scroll: This scroller is used in the ListView by default.

Container scroll: This scroller is used, when the height property of the ListView is set.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
    @Html.EJS().ListView("ui-
list").DataSource((IEnumerable<object>)ViewBag.listData).EnableVirtualizatio
n(true).Height(500).Render()
<style>
    #ui-list {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
        cursor: pointer;
    }
</style>

```

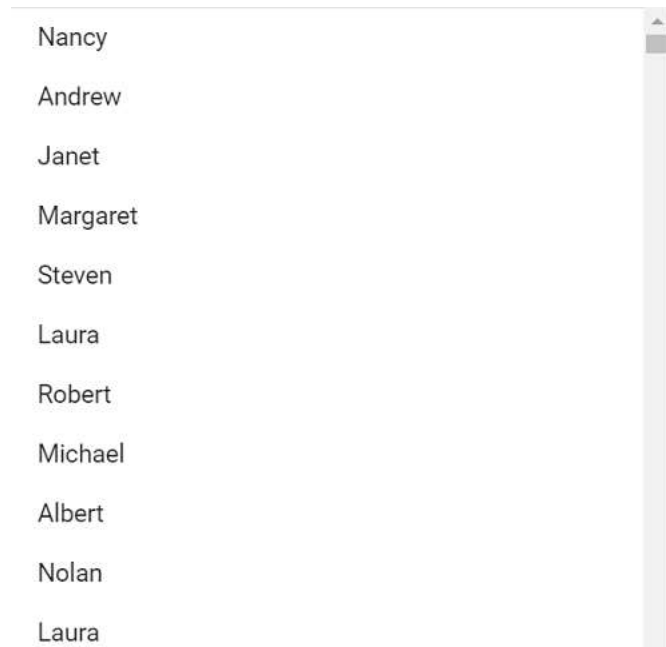
LISTCONTROLLER.CS

```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listData = new List<object>();
        listData.Add(new { text = "Nancy", id = "0" });
        listData.Add(new { text = "Andrew", id = "1" });
        listData.Add(new { text = "Janet", id = "2" });
        listData.Add(new { text = "Margaret", id = "3" });
        listData.Add(new { text = "Steven", id = "4" });
        listData.Add(new { text = "Laura", id = "5" });
        listData.Add(new { text = "Robert", id = "6" });
        listData.Add(new { text = "Michael", id = "7" });
        listData.Add(new { text = "Albert", id = "8" });
        listData.Add(new { text = "Nolan", id = "9" });
        for (int i = 10; i < 1000; i++)
        {
            int index = new Random().Next(0, 10);
            listData.Add(new
            {
                text =
listData[index].GetType().GetProperty("text").GetValue(listData[index],
null).ToString(),
                id = i.ToString()
            });
        }
        ViewBag.listData = listData;
        return View();
    }
}

```

Output be like the below.



We can use `template` property to customize list items in UI virtualization.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@{ var template = "<div class='list-container'><div id='icon'
class=\"${$imgUrl ? \"'img\" : \"icon\"}\">\" +
    \"<span class=\"${$imgUrl ? \"'hideUI\" : \"showUI\"}\">\" +
    \"${icon}</span> <img class=\"${$imgUrl ? \"'showUI\" : \"hideUI\"}\"
}\" width = 45 height = 45 src=\"${$imgUrl ? $imgUrl : \"\"}\" />\" +
    \"</div><div class='text'>${text}</div></div>\";
}
@Html.EJS().ListView("ui-
list").DataSource((IEnumerable<object>)ViewBag.listData).EnableVirtualizatio
n(true).ShowHeader(true).HeaderTitle("Contacts").Template(@template).Height(
500).Render()
<style>
    #ui-list {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
        cursor: pointer;
    }
    button {
        float: right
    }
    #icon {
        width: 45px;
        height: 45px;
        text-align: center;
        line-height: 45px;
        border-radius: 50%;
    }
```

```
    font-size: 20px;
    font-weight: 500;
    float: left;
    margin-top: 17px;
    margin-right: 35px;
}
img {
    border-radius: 50%;
    border: #ddd;
    border: 1px solid rgba(40, 40, 40, 0.12);
}
.R {
    background: purple;
}
.M {
    background: pink;
}
.A {
    background: green;
}
.S {
    background: lightskyblue;
}
.J {
    background: orange;
}
.N {
    background: blue;
}
#ui-list .e-list-item {
    height: 80px;
    border: #ddd;
    border: 1px solid rgba(184, 184, 184, 0.12);
}
.list-container {
    width: inherit;
    height: 100%;
}
.showUI {
    display: inline;
}
.hideUI {
    display: none;
}
.content {
    height: 100%;
    float: left;
}
.name {
    height: 100%;
    font-size: 20px;
    font-weight: 600;
    line-height: 78px;
}
</style>
```

LISTCONTROLLER.CS

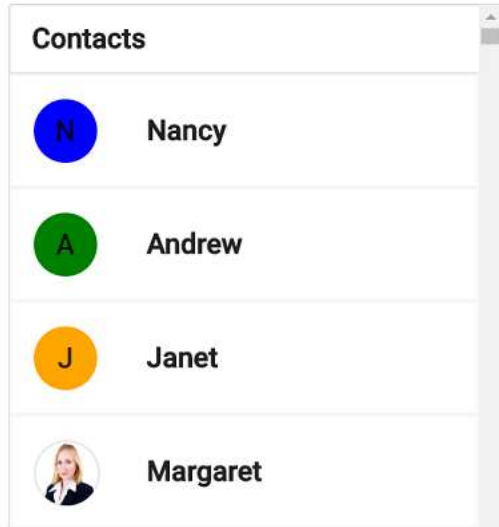
```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listData = new List<object>();
        listData.Add(new { text = "Nancy", imgUrl = "", icon = "N", id = "0"
    });
        listData.Add(new { text = "Andrew", imgUrl = "", icon = "A", id =
    "1" });
        listData.Add(new { text = "Janet", imgUrl = "", icon = "J", id = "2"
    });
        listData.Add(new { text = "Margaret", icon = "", imgUrl =
    "./margaret.png", id = "3" });
        listData.Add(new { text = "Steven", imgUrl = "", icon = "S", id =
    "4" });
        listData.Add(new { text = "Laura", icon = "", imgUrl =
    "./images/laura.png", id = "5" });
        listData.Add(new { text = "Robert", imgUrl = "", icon = "R", id =
    "6" });
        listData.Add(new { text = "Michael", imgUrl = "", icon = "M", id =
    "7" });
        listData.Add(new { text = "Albert", icon = "", imgUrl =
    "./images/albert.png", id = "8" });
        listData.Add(new { text = "Nolan", imgUrl = "", icon = "N", id = "9"
    });

        for (int i = 10; i < 1000; i++)
        {
            int index = new Random().Next(0, 10);
            listData.Add(new
            {
                text =
listData[index].GetType().GetProperty("text").GetValue(listData[index],
null).ToString(),
                icon =
listData[index].GetType().GetProperty("icon").GetValue(listData[index],
null).ToString(),
                imgUrl =
listData[index].GetType().GetProperty("imgUrl").GetValue(listData[index],
null).ToString(),
                id = i.ToString()
            });
        }
        ViewBag.listData = listData;
        return View();
    }
}

```

Output be like the below.



Conditional rendering

The following conditional rendering support is provided for the template and groupTemplate.

| Name | Syntax |

| --- | --- | --- |

| conditional class | `<div class="{ $id % 2 === 0 ? 'even-list' : 'odd-list' }"></div>` |

| conditional attribute | `<div id="{ $id % 2 === 0 ? 'even-list' : 'odd-list' }"></div>` |

| conditional text content | `<div>{ $id % 2 === 0 ? 'even-list' : 'odd-list' }</div>` |

In the following sample, the light blue is applied for the even list and light coral is applied for the odd list based on the conditional class.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@{ var template = "<div id='list-container' class='{ $id % 2 === 0 ?
\'even-list\' : \'odd-list\' }' > {text} </div>"; }
@Html.EJS().ListView("ui-
list").DataSource((IEnumerable<object>)ViewBag.listData).EnableVirtualizatio
n(true).Template(@template).Height(500).Render()
<style>
    #ui-list {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
        cursor: pointer;
    }
    #list-container{
        height: inherit;
        width: inherit;
        padding-left: 30px;
    }
    #ui-list .e-list-item{
```

```
padding: 0;
}
#ui-list .even-list{
    background-color: #cfd8dc;
}
#ui-list .odd-list{
    background-color: #eceff1;
}
</style>
```

LISTCONTROLLER.CS

```
public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listData = new List<object>();
        listData.Add(new { text = "Nancy", id = "0" });
        listData.Add(new { text = "Andrew", id = "1" });
        listData.Add(new { text = "Janet", id = "2" });
        listData.Add(new { text = "Margaret", id = "3" });
        listData.Add(new { text = "Steven", id = "4" });
        listData.Add(new { text = "Laura", id = "5" });
        listData.Add(new { text = "Robert", id = "6" });
        listData.Add(new { text = "Michael", id = "7" });
        listData.Add(new { text = "Albert", id = "8" });
        listData.Add(new { text = "Nolan", id = "9" });
        for (int i = 10; i < 1000; i++)
        {
            int index = new Random().Next(0, 10);
            listData.Add(new
            {
                text =
listData[index].GetType().GetProperty("text").GetValue(listData[index],
null).ToString(),
                id = i.ToString()
            });
        }
        ViewBag.listData = listData;
        return View();
    }
}
```

Output be like the below.



Scrolling in ASP.NET MVC Listview Control

Scrolling is a technique that allows you to load more items as the user scrolls through a list, providing a seamless and dynamic user experience.

Render the ListView with `dataSource`, and bind the `(scroll)`[\[https://help.syncfusion.com/cr/aspnetmvc-js2/Syncfusion.EJ2.Lists.ListView.html#SyncfusionEJ2ListsListViewScroll\]](https://help.syncfusion.com/cr/aspnetmvc-js2/Syncfusion.EJ2.Lists.ListView.html#SyncfusionEJ2ListsListViewScroll) event. Within the scroll event, you can access information such as the scroll direction, event name and the distance from the scrollbar to the top and bottom ends through the `distanceY` parameter.

In the given example, new data is seamlessly added while scrolling. When you scrolls to the bottom and the distance scrolled is less than 100 pixels, it dynamically loads a batch of items into the list as long as there are more items to render.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@{
    var template =
        "<div style='display: flex;' class='e-list-wrapper e-list-multi-line'>" +
        "<span style='display: block; white-space: normal; max-width: 80%; padding: 10px; background-color: #e0e0e0; border-radius: 10px; word-wrap: break-word; ' class='e-list-item-header'>${ text }</span>" +
        "</div>";
}
<div class="control-section">
    <div class="grid-container">
        <div>
            <h3>Chat</h3>

@Html.EJS().ListView("listview").DataSource((IEnumerable<object>)ViewBag.dataSource).Height("320px").Width("400px").Template(template).Scroll("onListScrolled").Render()
        </div>
    </div>
</div>
<style>
```

```

.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    grid-gap: 20px;
    align-items: center;
}
h3 {
    margin: 0;
}
/* Optional: Add styling to headings and divs */
.grid-container > div {
    border: 1px solid #ddd;
    padding: 20px;
    border-radius: 5px;
    background-color: #f4f4f4;
}
.e-listview{
    background-color: white;
}
</style>
<script>
    function onListScrolled(args) {
        var listObj_1 =
document.getElementById('listview').ej2_instances[0];
        var data = listObj_1.dataSource;
        var itemsPerScroll = 5;
        var result = [];
        if (args.scrollDirection === 'Bottom') {
            if (args.distanceY < 100) {
                if (initialData < data.length) {
                    var startIndex = initialData;
                    var endIndex = Math.min(initialData + itemsPerScroll,
data.length);

                    result = data.slice(startIndex, endIndex);
                    listObj_1.addItem(result);
                    initialData = endIndex;
                }
            }
        }
    }
}
</script>

```

LISTCONTROLLER.CS

```

public class ListViewController : Controller
{
    public IActionResult List()
    {
        List<object> listdata = new List<object>();
        listdata.Add(new
        {
            text= "Hi Guys, Good morning! \uD83D\uDE0A, I'm very delighted to
share with you the news that our team is going to launch a new mobile
application",
            positionClass= "right"
        });
    }
}

```

```

listdata.Add(new
{
    text= "Oh! That's great \uD83D\uDE42",
    positionClass= "left"
});
listdata.Add(new
{
    text= "That is a good news \uD83D\uDE00",
    positionClass= "right"
});
listdata.Add(new
{
    text="What kind of application we are gonna launch? \uD83E\uDD14",
    positionClass= "left"
});
listdata.Add(new
{
    text= "A kind of 'Emergency Broadcast App' like being able to just
invite someone to teams without it impacting how many people have official
access",
    positionClass= "right"
});
listdata.Add(new
{
    text= "Who will be the client users for our applications? ",
    positionClass= "left"
});
listdata.Add(new
{
    text= "Is currently the only way to invite someone through 0365?
Just wondering down the road how organization would want to handle that with
freelancers, like being able to just invite someone to teams without it
impacting how many people have official access \uD83D\uDE14",
    positionClass= "right"
});
listdata.Add(new
{
    text= "Yes, however, that feature of inviting someone from outside
your organization is planned - expected closer to GA next year
\uD83D\uDC4D",
    positionClass= "left",
});
listdata.Add(new
{
    text= "I guess we should switch things over to hear for a while. How
long does the trial last? \uD83E\uDD14",
    positionClass= "right",
});
listdata.Add(new
{
    text = "I think the trial is 30 days. \uD83D\uDE03",
    positionClass = "left"
});
listdata.Add(new
{

```



```

        text = "Only 0365 only members of your organization. They said that  

        they are listening to customer feedback and hinted that guest users would be  

        brought in down the road \uD83D\uDE09",  

        positionClass = "right"  

    });  

    listdata.Add(new  

    {  

        text = "Cool thanks! \uD83D\uDC4C",  

        positionClass = "left"  

    });  

    var initialData = listdata.Take(5).ToList();  

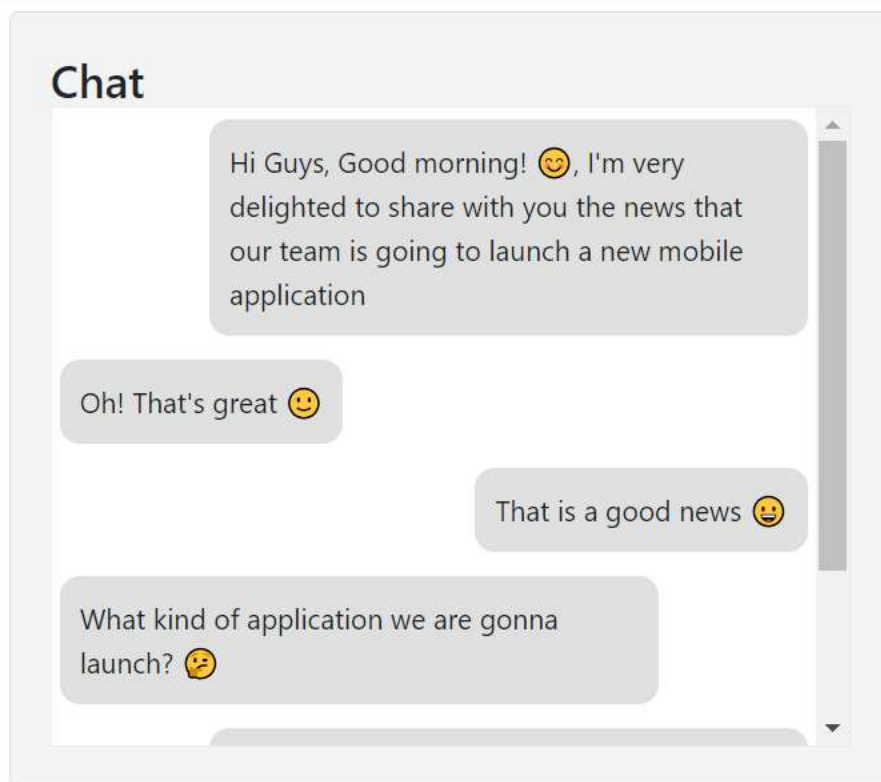
    ViewBag.dataSource = initialData;  

    return View();  

}

```

Output be like the below.



CSS Structure

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

Customizing ListView

Use the following CSS to customize the ListView.

`CSS

```
.e-listview {
```

```
border: 5px solid rgb(173, 255, 47);  
}
```

,

Customizing the list items

Use the following CSS to customize the items of ListView.

```
`CSS
```

```
.e-listview .e-list-item {  
text-align: center;  
color: pink;  
background-color: #2fa1ff;  
}
```

,

Customizing ListView's header

Use the following CSS to customize the header of ListView control.

```
`CSS
```

```
.e-listview .e-list-header{  
color: #2fa1ff;  
justify-content: center;  
}
```

,

Customizing group header of ListView

Use the following CSS to customize the category of the group items.

```
`CSS
```

```
.e-listview .e-list-group-item {  
color: rgb(173, 255, 47);  
background-color: maroon;  
text-align: end;  
}
```

,

Customizing the hover state of ListView control

Use the following CSS to customize the list item when hovering.

Customizing ListView hover state with the checkbox checked

```
`CSS
```

```
.e-listview .e-list-item.e-hover.e-active.e-checklist {
```

```
color: rgb(83, 5, 79);
background-color: rgb(173, 255, 47);
}
,
```

Customizing ListView hover state

```
`CSS
.e-listview .e-list-item.e-hover {
color:red;
background-color: rgb(173, 255, 47);
}
,
```

Customizing selected item of ListView control

Use the following CSS to customize the selected list item.

Customizing ListView's selected item with the checkbox checked

```
`CSS
.e-listview .e-list-item.e-checklist.e-focused.e-active {
color: rgb(83, 5, 79);
background-color: rgb(0, 15, 100);
}
,
```

Customizing ListView's selected item

```
`CSS
.e-listview .e-list-item.e-focused {
color: #2fa1ff;
background-color: rgb(0, 15, 100);
}
,
```

Accessibility in ASP.NET MVC ListView component

The ListView component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the ListView component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The ListView component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the ListView component:

- | Attributes | Purpose |
- | --- | --- |
- | **role=list** | Specifies the non selectable list container. |
- | **role=listitem** | Specifies the role of each item in a selectable ListView and its containment within the list. |
- | **role=presentation** | Specifies the role of non selectable list element. |

- | `role=checkbox` | Indicates checkbox control along with listitem element. |
- | `aria-checked` | Indicates the current checked state of checkbox. |
- | `aria-label` | Provides an accessible name for the ListView Checkbox. |
- | `aria-disabled` | Indicates element is perceivable but disabled. |

Keyboard interaction

The ListView component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the ListView component.

| Keyboard shortcuts | Actions |

|-----|-----|

| Arrow Up | Move to the previous list item |

| Arrow Down | Move to the next list item |

| Space | Check and uncheck the targeted list from the whole list |

| BackSpace | Get back to the previous lists if it is nested list |

| Home | Moves focus to first list item |

| End | Moves focus to last list item |

Ensuring accessibility

The ListView component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the ListView component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the ListView component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Migration from Essential JS 1

This article describes the API migration process of ListView component from Essential JS 1 to Essential JS 2.

| Behaviour | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Virtualization | **Property:** `AllowVirtualScrolling`

`@Html.EJ().ListView("list").AllowVirtualScrolling(true)
.VirtualScrollMode(VirtualScrollMode.Continuous)` | **Property:** `EnableVirtualization`

`@Html.EJS().ListView("list").DataSource((IEnumerable<object>)
ViewBag.dataSource).EnableVirtualization(true).Render()` |

| Fields | **Property:** `FieldSettings`

 `@Html.EJ().ListView("list").FieldSettings(e => e.Text("text"))` | **Property:** `Fields`

 Inner properties: `child, enabled, groupBy`

htmlAttributes, iconCss, id, isChecked etc.


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Fields(new Syncfusion.EJ2.Lists.ListViewFieldSettings { GroupBy="type"}).Render() |
```

| Template | **Property:** *RenderTemplate*


```
@Html.EJ().ListView("list").RenderTemplate(true)<br/>.TemplateId("template") <br/> <br/>
```

```
<div id="template"><div>Data 1</div><div>Data 2</div><div>Data 3</div> </div> | Property:
```

```
Template <br/><br/> @ { var template = "<div class='template'>${text}</div>"; } <br/><br/>
```

```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Template(template).Render() |
```

| Animation | **Not Applicable** | **Property:** *Animation*

 List<object> animation = new List<object>();
 animation.Add(new { effect = "SlideLeft", duration = "400", easing = "ease" });


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Animation(ViewBag.animation).Render() |
```

| Enable | **Not Applicable** | **Property:** *Enable*


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Enable(true).Render() |
```

| Template for grouping | **Not Applicable** | **Property:** *GroupTemplate*

 @ { var groupTemplate = "<div class='template'>\${text}</div>"; }


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).GroupTemplate(groupTemplate).Render() |
```

| Template for header | **Not Applicable** | **Property:** *HeaderTemplate*

 @ { var headerTemplate = "<div class='template'>\${text}</div>"; }


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).HeaderTemplate(headerTemplate).Render() |
```

| HTML attributes | **Not Applicable** | **Property:** *HtmlAttributes*

 @ {
IDictionary<string, object> htmlAttribute = new Dictionary<string, object>();
htmlAttribute.Add("class", "listViewCustom");
}


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).HtmlAttributes(htmlAttribute).Render() |
```

| Clear | **Method:** *clear()*

 @Html.EJ().ListView("list").DataSource(ViewBag.data)

 var listObj = \$("#list").ejListView("instance");
 listObj.clear(); | **Property** *clear*


```
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Render() <br/><br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/> listObj.clear(); |
```

| isChecked | **Method:** *isChecked()* @Html.EJ().ListView("list").EnableCheckMark(true) var listObj = \$("#list").ejListView("instance"); listObj.isChecked(); | **Not Applicable** |

| Load Ajax Content | **Property:** *EnableAjax*


```
@Html.EJ().ListView("list").EnableAjax(true).Items(items => items.Add().Text("Data
```

```

1").Href("/html/template.html")) | Event: onSuccess <br/><br/>
@Html.EJS().ListView("list").ActionBegin("onBegin").Render() <br/> <br/> function onBegin() {
<br/> var ajax = new ej.base.Ajax("/html/template.html", "GET", false); <br/> ajax.onSuccess =
function (value) { <br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/>
listObj.template = value; <br/> listObj.dataSource = @Json.Serialize(ViewBag.data); <br/> }
<br/> ajax.send(); <br/> }|

| Remove CheckMark | Method: removeCheckMark() <br/><br/>
@Html.EJ().ListView("list").EnableCheckMark(true) <br/><br/> var listObj =
$("#list").ejListView("instance"); <br/> listObj.removeCheckMark(2); | Method: uncheckItem()
<br/><br/>
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Render()
<br/><br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/>
listObj.uncheckItem({ id:'2' });|

| Set Active | Method: setActive() <br/><br/>
@Html.EJ().ListView("list").DataSource(ViewBag.data) <br/><br/> var listObj =
$("#list").ejListView("instance"); <br/> listObj.setActive(2); | Method: selectItem() <br/><br/>
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Render()
<br/><br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/>
listObj.selectItem({ id:'2' });|

| Select | Not Applicable | Event: select <br/><br/>
@Html.EJS().ListView("list").DataSource((IEnumerable<object>)<br/>ViewBag.dataSource).Select(
"onSelect").Render() <br/> <br/> function onSelect() { } |

| Ajax Before Load | Event: AjaxBeforeLoad <br/><br/>
@Html.EJ().ListView("list").EnableAjax(true).Items(items => items.Add().Text("Data
1").Href("/html/template.html")).ClientSideEvents(e => e.AjaxBeforeLoad("onAjaxBefore"))
<br/> <br/> function onAjaxBefore() { } | Event: beforeSend <br/><br/>
@Html.EJS().ListView("list").ActionBegin("onBegin").Render() <br/> <br/> function onBegin() {
<br/> var ajax = new ej.base.Ajax("/html/template.html", "GET", false); <br/> ajax.onSuccess =
function (value) { <br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/>
listObj.template = value; <br/> listObj.dataSource = @Json.Serialize(ViewBag.data); <br/> }
<br/> ajax.beforeSend = function (value) { } <br/> ajax.send(); <br/> } |

| Ajax Complete | Event: AjaxComplete <br/><br/>
@Html.EJ().ListView("list").EnableAjax(true).Items(items => items.Add().Text("Data
1").Href("/html/template.html")).ClientSideEvents(e => e.AjaxComplete("onAjaxComplete"))
<br/> <br/> function onAjaxComplete() { } | Event: onSuccess <br/><br/>
@Html.EJS().ListView("list").ActionBegin("onBegin").Render() <br/> <br/> function onBegin() {
<br/> var ajax = new ej.base.Ajax("/html/template.html", "GET", false); <br/> ajax.onSuccess =
function (value) { <br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/>
listObj.template = value; <br/> listObj.dataSource = @Json.Serialize(ViewBag.data); <br/> }
<br/> ajax.send(); <br/> }|

| Ajax Error | Event: AjaxError <br/><br/>
@Html.EJ().ListView("list").EnableAjax(true).Items(items => items.Add().Text("Data

```

```
1").Href("/html/template.html")).ClientSideEvents(e => e.AjaxError("onAjaxError")) <br/> <br/>
function onAjaxError() { } | Event: onError <br/><br/>
@Html.EJS().ListView("list").ActionBegin("onBegin").Render() <br/> <br/> function onBegin() {
<br/> var ajax = new ej.base.Ajax("/html/template.html", "GET", false); <br/> ajax.onSuccess =
function (value) { <br/> var listObj = document.getElementById('list').ej2_instances[0]; <br/>
listObj.template = value; <br/> listObj.dataSource = @Json.Serialize(ViewBag.data); <br/> }
<br/> ajax.onError = function (value) { } <br/> ajax.send(); <br/> }|
```

How To

Get selected items from ListView

Single or many items can be selected by users in the ListView component. An API is used to get selected items from the list items. This is called as the [getSelectedItems](#) method.

getSelectedItems method

This is used to get the details of the currently selected item from the list items. It returns the [SelectedItem](#) | [SelectedCollection](#)

The [getSelectedItems](#) method returns the following items from the selected list items.

| Return type | Purpose |
|-------------|--|
| ----- ----- | |
| text | Returns the text of selected item lists |
| data | Returns the whole data of selected list items, i.e., returns the fields data of selected li. |
| item | Returns the collections of list items |

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).Fields(new ListViewFieldSettings { Text = "text",
Id="id"}).ShowCheckBox(true).Render()
    <div id="text">
        @Html.EJS().Button("btn").Content("Get selected Items").Render()
        <table id="val"></table>
    </div>
<style>
    #text {
        margin-left: 10px;
        margin-top: 20px;
    }
    #element {
        max-width: 350px;
        margin: auto;
        margin-top: 10px;
        display: block;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
<script>
```



```

document.getElementById('btn').addEventListener('click', () => {
    var selecteditem =
document.getElementById('element').ej2_instances[0].getSelectedItems();
    var data = document.getElementById('val');
    data.innerHTML = "";
    var row1 = document.createElement('tr');
    var header1 = document.createElement('th');
    header1.innerHTML = 'Text';
    row1.appendChild(header1);
    var header2 = document.createElement('th');
    header2.innerHTML = 'Id';
    row1.appendChild(header2);
    document.getElementById('val').appendChild(row1);
    for (var i= 0; i < (selecteditem["data"]).length; i++) {
        var row2 = document.createElement('tr');
        row2.setAttribute("id", i.toString());
        var data1 = document.createElement('td');
        data1.innerHTML = selecteditem["text"][i];
        row2.appendChild(data1);
        var data2 = document.createElement('td');
        data2.innerHTML = (selecteditem["data"][i].id.toString());
        row2.appendChild(data2);
        document.getElementById('val').appendChild(row2);
    }
});
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new
            {
                text = "Hennessey Venom",
                id = "list-01"
            }); listdata.Add(new
            {
                text = "Bugatti Chiron",
                id = "list-02"
            }); listdata.Add(new
            {
                text = "Bugatti Veyron Super Sport",
                id = "list-03"
            }); listdata.Add(new
            {
                text = "SSC Ultimate Aero",

```

```

        id = "list-04"
    }); listdata.Add(new
    {
        text = "Koenigsegg CCR",
        id = "list-05"
    }); listdata.Add(new
    {
        text = "McLaren F1",
        id = "list-06"
    }); listdata.Add(new
    {
        text = "Aston Martin One- 77",
        id = "list-07"
    }); listdata.Add(new
    {
        text = "Jaguar XJ220",
        id = "list-08"
    });
    ViewBag.dataSource = listdata;
    return View();
}
}
}

```

Use Dynamic templates in ListView based on device

The Syncfusion Essential JS2 components are desktop and mobile-friendly. So, you can use Syncfusion components in both modes. The component templates are not always fixed. Applications may need to load various templates depending upon the device.

Integration

In the ListView component, template support is being used. In some cases, the component wrapper is always responsive across all devices, but the template contents are dynamically changed with unspecified (sample side) dimensions. CSS customization is also needed in sample-side to align template content responsively in both mobile and desktop modes. Here, two templates have been loaded for mobile and desktop modes. To check the device mode, a **browser module** has been imported from the [ej2-base](#) package.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@{
    var wintemplate = "";
    var width = "";
    var mob_template = "<div class='settings'>"
        + "<div id='postContainer'><div id='postImg'> "
        + "<img src=${image} /></div>"
        + "<div id='content'> "
        + "<div id='info'> "
        + "<div id='logo'> <div id= 'share'> "
        + "<span class='share'></span> </div> <div id='comments'>
        <span class='comments'></span> </div>"
        + "<div id='bookmark'> <span class='bookmark'></span>
        </div></div></div>"
        + "<div class='name'>${Name}</div>"
    }

```

```

        + "<div class='description'>${content}</div>"
        + "<div class='timeStamp'>${timeStamp} </div>"
        + "</div>";
    var template = "<div class='settings'><div id='postContainer'><div
id='postImg'><img src=${image} /></div><div id='content'>"
        + "<div class='name'>${Name}</div><div
class='description'>${content}</div><div id='info'>"
        + "<div id='logo'><div id='share'><span
class='share'></span>"
        + "</div> <div id='comments'> <span
class='comments'></span> </div>"
        + "<div id='bookmark'> <span
class='bookmark'></span> </div></div>"
        + "<div class='timeStamp'>${timeStamp}
</div></div></div>";
    if (Request.Browser.IsMobileDevice)
    {
        width = "350px";
        wintemplate = mob_template;
    }
    else
    {
        wintemplate = template;
    }
}
@Html.EJS().ListView("List").DataSource((IEnumerable<object>)ViewBag.dataSou
rce).HeaderTitle("Syncfusion
Blog").ShowHeader(true).Width(width).Template(wintemplate).Render()
<style>
    #List img {
        height: 70px;
        width: 70px;
        border-radius: 50%;
        border: 1px solid #ccc;
    }
    #List {
        display: block;
        max-width: 400px;
        margin: auto;
        border: 1px solid;
        border-color: #c3c3c3;
        border-color: rgba(0, 0, 0, 0.12);
    }
    #List .settings {
        height: 170px;
        line-height: 70px;
        margin-left: 16px;
        margin-right: 16px;
    }
    #List .e-list-item {
        height: 170px;
        line-height: 70px;
        padding: 0;
        cursor: pointer;
        border-bottom: 0.8px solid #ddd;
    }
    #time {

```

```
        line-height: 23px;
        margin-top: 13px;
        padding-left: 10px;
        width: 191px;
    }
    #img {
        float: left;
        padding-top: 6px;
        padding-left: 0;
        padding-right: 20px;
    }
    #info {
        padding-top: 3px;
    }
    #List .e-list-header {
        color: white;
        height: 57px;
        background-color: #56697f;
        padding-left: 15px;
        position: relative;
        top: -1px;
    }
    .bootstrap #List .e-list-header .e-text {
        line-height: 18px;
    }
    #List .e-list-header .e-text {
        font-family: sans-serif;
        font-size: 18px;
        line-height: 26px;
    }
    #List .category {
        font-family: "serif";
        font-weight: 600;
        font-size: 19px;
    }
    #List .name {
        font-size: 15px;
        font-weight: 500;
        line-height: 23px;
    }
    #List .content {
        line-height: 19px;
        font-size: 13px;
        font-weight: 200;
    }
    #List .e-hover {
        transition: 0.5s;
    }
    .timeStamp {
        font-size: small;
        margin-bottom: 1px;
        margin-top: -17px;
    }
    .bookmark::before {
        color: grey;
        float: right;
        line-height: 0;
```

```

        margin-bottom: 1px;
        font-family: "Bookmarks";
        content: "\e700";
        margin-left: 3px;
        margin-right: 3px;
        font-size: 16px;
        padding-top: 19px;
        padding-bottom: 5px;
    }
    .share::before {
        color: grey;
        float: right;
        line-height: 0;
        margin-bottom: 1px;
        font-family: "Bookmarks";
        content: "\e701";
        margin-left: 3px;
        margin-right: 3px;
        font-size: 13px;
        padding-top: 19px;
        padding-bottom: 5px;
    }
    .comments::before {
        color: grey;
        float: right;
        line-height: 0;
        margin-bottom: 1px;
        font-family: "Bookmarks";
        content: "\e703";
        margin-left: 3px;
        margin-right: 3px;
        font-size: 15px;
        padding-top: 19px;
        padding-bottom: 5px;
    }
    .bookmark {
        cursor: pointer;
    }
    .share {
        cursor: pointer;
    }
    .comments {
        cursor: pointer;
    }
    .description {
        color: grey;
        line-height: 20px;
        font-size: 15px;
        font-weight: 200;
        text-align: justify;
    }
    /* csslint ignore:start */
    @@font-face {
        font-family: 'Bookmarks';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAwAgTlMvMj0gSRgAAAEoAAAAVmNtYXDOl85qAAABkAAAAEJnbHlmRXC
i8wAAAEAAAAFkaGVhZA8SahsAAADQAAAAANmhoZWEHmQNTAAAArAAAACRobXR4D7gAAAAAAAYAAAAA

```

```

QbG9jYQDwAIAAAAHUAAAAACm1heHABEQAYAAABCAAAACBuYW1lFuNPLwAAA0QAAAI9cG9zdLaVZAw
AAAWAAAAAXQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAABAABAAAAQAAGHTc9V8
PPPUACwPoAAAAANYFEqYAAAAA1gUSPgAAAAAD6gPqAAAACAACAAAAAAAAAAAAEACyAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPuAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAAAAAAIAAADAAAAFAADAAEAAAAUAAQALgAAAAAYABAABAAALnAecD//8AA0cA5wP//wA
AAAAAAQAGAAgAAAAABAAIAAwAAAAAAAAA+AIASgAAAAMAAAAAAxD6gANABkAJQAAExE3FxEHLgE
nNDcjDgElMxUzFSMVIZUjNTMHHgEXPgE3LgEnDgHQ190MWXcCCWU0RAGWKFBQKFBQ1QJdRkZdAQF
dRkZdAwn8+fn5AnMBAndZHx0BRWhQKFBQKA5GXQICXUZGXQEBXQAAAAABAAAAAPqA+oAJAAACQE
uASMOAQceARcyNjCBHgEXPgE3LgIHCQEWmZ4BNy4BJw4BArn+QxM1HD5WAgJtQRwyEwHGC1I5P1U
BAVOCKf5YAbUmND5WAQFWPkJUA2T+7hESAko3OUwBEQ7+6zJAAGJLOtPLASUBBgEMHAFLOtPLAQF
LAAACAAAAAPqA4EADwAcAAABHgEXMjcxJz4BNS4BJw4BBTMVNzMnJjU+ATc1IQIOA4Z1FROGLzM
8AoZmZYb98YWBygIRBLOG/QYBvGWHAgRmhyBpQGWGAwOG0sLCBzA2h7MDiAAAAAASAN4AAQAAAA
AAAAABAAAAQAAAAAAQAJAAEAQAQAAAAAAGAHAAoAAQAAAAAAAwAJABEAQAQAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACgAsADCAAQAAAAAACwASAGMAAwABBAkAAAAACAHU
AAwABBAkAAQASAHCAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgQm9va21hcmtzUmVndWxhckJ
vb2ttYXJrc0Jvb2ttYXJrc1ZlcnNpb24gMS4wQm9va21hcmtzRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Z1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABCAG8AbwBrAG0AYQB
yAGsAcwBSAGUAZwB1AGwAYQByAEIAbwBvAGsAbQBhAHIAawBzAEIAbwBvAGsAbQBhAHIAawBzAFY
AZQByAHMAaQBvAG4AIAAxAC4AMABCAG8AbwBrAG0AYQByAGsAcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIABlAHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAQIBAwEEAQUADGJvb2ttYXJrLWfKzApzaGFyZS0
tLTaxF21lc3NhZ2VzLWluZm9ybWF0aW9uLTaxAAAAAA=) format('truetype');
    font-weight: normal;
    font-style: normal;
}
/* csslint ignore:end */
#postImg {
    margin-right: 25px;
    margin-top: 30px;
}
#postContainer {
    width: inherit;
    margin-top: 10px;
    display: inline-flex;
}
</style>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new
            {

```

```

        Name = "IBM Open-Sources Web Sphere Liberty Code",
        content = "In September, IBM announced that it would be
open-sourcing the code for WebSphere...",
        id = "1",
        image =
"https://ej2.syncfusion.com/demos/src/listview/images/1.png",
        timeStamp = "Syncfusion Blog - October 19, 2017"
    });
    listdata.Add(new
    {
        Name = "Must Reads: 5 Big Data E-books to upend your
development",
        content = "Our first e-book was published in May 2012-jQuery
Succinctly was the start of over...",
        id = "2",
        image =
"https://ej2.syncfusion.com/demos/src/listview/images/2.png",
        timeStamp = "Syncfusion Blog - October 18, 2017"
    });
    listdata.Add(new
    {
        Name = "The Syncfusion Global License: Your Questions,
Answered ",
        content = "Syncfusion recently hosted a webinar to cover the
ins and outs of the Syncfusion global...",
        id = "4",
        image =
"https://ej2.syncfusion.com/demos/src/listview/images/3.png",
        timeStamp = "Syncfusion Blog - October 18, 2017"
    });
    listdata.Add(new
    {
        Name = "Know: What is Coming from Microsoft this Fall ",
        content = "On October 17, Microsoft will release its Fall
Creators Update for the Windows 10 platform...",
        id = "5",
        image =
"https://ej2.syncfusion.com/demos/src/listview/images/6.png",
        timeStamp = "Syncfusion Blog - October 17, 2017"
    });
    ViewBag.dataSource = listdata;
    return View();
}
}
}

```

Create Dual List from ListView

The dual list contains two ListView. This allows you to move list items from one list to another using the client-side events. This section explains how to integrate the ListView component to achieve dual list.

Use cases

- Stock exchanges of two different countries
- Job applications (skill sets)

Integration of Dual List

Here, two ListView components have been used to display the list items. An ej2-button is used to transfer data between the ListView, and a textbox is used to achieve the UI of filtering support.

The dual list supports:

- Moving whole data from one list to another.
- Moving selected data from one list to another.
- Filtering the list by using a client-side typed character.

In the ListView component, sorting is enabled using the [sortOrder](#) property, and the [select](#) event is triggered while selecting an item. Here, the select event is triggered to enable and disable button states.

Manipulating data

Moving whole data from the first list to the second list(>>)

- Here, the whole data can be moved from the first ListView to the second by clicking the first button. When clicking the button, the whole list items are sliced, and `concat` with the second ListView. This button is enabled only when the data source of the first ListView is not empty.

Moving whole data from the second list to the first list(<<)

- The functionality of the second button is the same as above, and data is transferred from the second list to the first list. This button is enabled only when the data source of the second ListView is not empty.

Moving selected item from one list to another list (>) and (<)

- The [Select](#) event is triggered when selecting a list item in the ListView. The selected items can be transferred between two lists. These buttons will be enabled when selecting an item in lists.

Filtering method

- The filtering method is used to filter list items when typing a character in the text box. In this method, the [dataManager](#) has been used to fetch data from the data source and display in ListView.

Sorting

- By using the dual list, list items can be sorted in the ListView component using the [sortOrder](#) property.

You can enable sorting in one ListView; in the same order, data can be transferred to another ListView.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
<div class="list_container">
    <div id="list_container_1">
```



```

<input class="e-input" type="text" id="firstInput"
placeholder="Filter" title="Type in a name">
    @Html.EJS().ListView("list-
1").DataSource((IEnumerable<object>)ViewBag.firstListData).Select("onFirstLi
stSelect").SortOrder(Syncfusion.EJ2.Lists.SortOrder.Ascending).Render()
</div>
<div id="btn">
    @Html.EJS().Button("firstBtn").Content(">>").Render()
    @Html.EJS().Button("secondBtn").Content(">").Disabled(true).Render()
    @Html.EJS().Button("thirdBtn").Content("<").Disabled(true).Render()
    @Html.EJS().Button("fourthBtn").Content("<<").Render()
</div>
<div id="list_container_2">
    <input class="e-input" type="text" id="secondInput"
placeholder="Filter" title="Type in a name">
    @Html.EJS().ListView("list-
2").DataSource((IEnumerable<object>)ViewBag.secondListData).Select("onSeconL
istSelect").SortOrder(Syncfusion.EJ2.Lists.SortOrder.Ascending).Render()
</div>
</div>
<style>
.list_container {
    height: 398px;
    max-width: 485px;
    margin: auto;
}
#list_container_1,
#list_container_2 {
    width: 200px;
}
#list-1,
#list-2 {
    height: 362px;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#list_container_1,
#list_container_2 {
    display: inline-block;
}
.e-btn {
    margin-bottom: 15px;
    width: 40px;
    height: 40px;
}
#btn {
    display: inline-block;
    width: 41px;
    margin: 0 15px;
    position: relative;
    top: -53%;
    transform: translateY(50%);
}
</style>
<script type="text/javascript">
    var firstListObj, secondListObj, firstBtnObj, secondBtnObj, thirdBtnObj,
fourthBtnObj, firstListData, secondListData;

```

```

window.onload = function () {
    firstListObj = document.getElementById("list-1").ej2_instances[0];
    secondListObj = document.getElementById("list-2").ej2_instances[0];
    secondBtnObj =
document.getElementById("secondBtn").ej2_instances[0];
    thirdBtnObj = document.getElementById("thirdBtn").ej2_instances[0];
    firstBtnObj = document.getElementById("firstBtn").ej2_instances[0];
    fourthBtnObj =
document.getElementById("fourthBtn").ej2_instances[0];
    firstListData = firstListObj.dataSource.slice();
    secondListData = secondListObj.dataSource.slice();
}
//Here we are moving all list items to second list on clicking move all button
document.getElementById("firstBtn").addEventListener('click', function
() {
    secondListObj.dataSource =
Array.prototype.concat.call(firstListObj.dataSource,
secondListObj.dataSource);
    secondListObj.dataBind();
    updateFirstListData();
    firstListObj.removeMultipleItems(firstListObj.liCollection);
    firstListData = firstListData.concat(firstListObj.dataSource);
    secondListData = secondListObj.dataSource.slice();
    firstBtnObj.disabled = true;
    onFirstKeyUp();
    setButtonState();
});
//Here we are moving selected list item to second list on clicking move button
document.getElementById("secondBtn").addEventListener('click', function
() {
    var e = firstListObj.getSelectedItems();
    secondListObj.dataSource =
Array.prototype.concat.call(secondListObj.dataSource, e.data);
    secondListObj.dataBind();
    updateFirstListData();
    firstListObj.removeItem(e.item);
    firstListData = firstListData.concat(firstListObj.dataSource);
    secondListData = secondListObj.dataSource.slice();
    onFirstKeyUp();
    secondBtnObj.disabled = true;
    setButtonState();
});
//Here we are moving selected list item to first list on clicking move button
document.getElementById("thirdBtn").addEventListener('click', function
() {
    var e = secondListObj.getSelectedItems();
    firstListObj.dataSource =
Array.prototype.concat.call(firstListObj.dataSource, e.data);
    firstListObj.dataBind();
    updateSecondListData();
    secondListObj.removeItem(e.item);
    secondListData = secondListData.concat(secondListObj.dataSource);
    firstListData = firstListObj.dataSource.slice();
    onSecondKeyUp();

```

```

        thirdBtnObj.disabled = true;
        setButtonState();
    });
    //Here we are moving all list items to first list on clicking move all button
    document.getElementById("fourthBtn").addEventListener('click', function
    () {
        firstListObj.dataSource =
        Array.prototype.concat.call(firstListObj.dataSource,
        secondListObj.dataSource);
        firstListObj.dataBind();
        updateSecondListData();
        secondListObj.removeMultipleItems(secondListObj.liCollection);
        secondListData = secondListData.concat(secondListObj.dataSource);
        firstListData = firstListObj.dataSource.slice();
        onSecondKeyUp();
        setButtonState();
    });
    //Here we are updating ListView dataSource for First List
    function updateFirstListData() {
        Array.prototype.forEach.call(firstListObj.liCollection, function
    (list) {
            firstListData.forEach(function (data, index) {
                if (list.innerText.trim() === data.text) {
                    delete firstListData[index];
                }
            });
        });
        document.getElementById("firstInput").value = '';
        var ds = [];
        firstListData.forEach(function (data) {
            ds.push(data);
        })
        firstListData = ds;
    }
    //Here we are updating ListView dataSource for second List
    function updateSecondListData() {
        Array.prototype.forEach.call(secondListObj.liCollection, function
    (list) {
            secondListData.forEach(function (data, index) {
                if (list.innerText.trim() === data.text) {
                    delete secondListData[index];
                }
            });
        });
        document.getElementById("secondInput").value = '';
        var ds = [];
        secondListData.forEach(function (data) {
            ds.push(data);
        })
        secondListData = ds;
    }
    function onFirstListSelect() {
        secondBtnObj.disabled = false;
    }
    function onSeconListSelect() {
        thirdBtnObj.disabled = false;
    }

```

```

    }
    //Here we are handling filtering of list items using dataManager for
    first List
    function onFirstKeyUp(e) {
        var value = document.getElementById("firstInput").value;
        var data = new ej.data.DataManager(firstListData).executeLocal(new
ej.data.Query().where('text', 'startswith', value, true));
        if (!value) {
            firstListObj.dataSource = firstListData.slice();
        } else {
            firstListObj.dataSource = data;
        }
        firstListObj.dataBind();
    }
    //Here we are handling filtering of list items using dataManager for
    second List
    function onSecondKeyUp(e) {
        var value = document.getElementById("secondInput").value;
        var data = new ej.data.DataManager(secondListData).executeLocal(new
ej.data.Query().where('text', 'startswith', value, true));
        if (!value) {
            secondListObj.dataSource = secondListData.slice();
        } else {
            secondListObj.dataSource = data;
        }
        secondListObj.dataBind();
    }
    //Here we are changing the button state
    function setButtonState() {
        if (firstListObj.dataSource.length) {
            firstBtnObj.disabled = false;
        } else {
            firstBtnObj.disabled = true;
            secondBtnObj.disabled = true;
        }
        if (secondListObj.dataSource.length) {
            fourthBtnObj.disabled = false;
        } else {
            fourthBtnObj.disabled = true;
            thirdBtnObj.disabled = true;
        }
    }
}
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
    }
}

```

```
{
    List<object> listdata = new List<object>();
    listdata.Add(new
    {
        text = "Hennessey Venom",
        id = "list-01"
    }); listdata.Add(new
    {
        text = "Bugatti Chiron",
        id = "list-02"
    }); listdata.Add(new
    {
        text = "Bugatti Veyron Super Sport",
        id = "list-03"
    }); listdata.Add(new
    {
        text = "SSC Ultimate Aero",
        id = "list-04"
    }); listdata.Add(new
    {
        text = "Koenigsegg CCR",
        id = "list-05"
    }); listdata.Add(new
    {
        text = "McLaren F1",
        id = "list-06"
    });
    List<object> listdata1 = new List<object>();
    listdata1.Add(new
    {
        text = "Aston Martin One- 77",
        id = "list-07"
    });
    listdata1.Add(new
    {
        text = "Jaguar XJ220",
        id = "list-08"
    });
    listdata1.Add(new
    {
        text = "McLaren P1",
        id = "list-09"
    });
    listdata1.Add(new
    {
        text = "Ferrari LaFerrari",
        id = "list-10"
    });
    ViewBag.firstListData = listdata;
    ViewBag.secondListData = listdata1;
    return View();
}
}
```

Load List Items in child list dynamically

To load list items in child list dynamically, push the new list item data into the existing [dataSource](#) using the [select](#) event.

Refer to the following steps to load list item into the child list:

1. Initially, render the ListView with the required data source.
2. Bind the [select](#) event that triggers selecting list item in the ListView component. By using the select event, you can push the new list item to the child list of the data source on specifying its item index. Item index can be obtained from the [SelectEventArgs](#) of the select event.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
@Html.EJS().ListView("listview").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { IconCss = "icon", Tooltip = "text"
}).Select("onSelect").ShowIcon(true).ShowHeader(true).HeaderTitle("Folders")
.Render()
<style>
    #listview {
        display: block;
        max-width: 350px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #listview.e-listview .e-list-icon {
        height: 24px;
        width: 30px;
    }
    .folder, .file {
        background:
            url('https://ej2.syncfusion.com/demos/src/listview/images/file_icons.png')
            no-repeat;
        background-size: 302%;
    }
    .folder {
        background-position: -5px -466px;
    }
    .file {
        background-position: -5px -151px;
    }
    /* csslint ignore:start */
    .new-list {
        color: deeppink !important;
    }
    /* csslint ignore:end */
</style>
<script>
    //Select event to add new list item in child page
    function onSelect(args) {
        //Add new file to the child page of selected list item
```

```

        this.dataSource[args.index].child.push({ id: '01-02', text: 'Newly
Added File', icon: 'file', htmlAttributes: { role: 'li', class: 'new-list' }
});
    }
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new
            {
                id = "01",
                text = "Music",
                icon = "folder",
                child = new List<object>() { new { id = "01-01", text =
"Gouttes.mp3", icon = "file" } }
            });
            listdata.Add(new
            {
                id = "02",
                text = "Videos",
                icon = "folder",
                child = new List<object>() {
                    new { id= "02-01", text= "Naturals.mp4", icon= "file" },
                    new { id= "02-02", text= "Wild.mpeg", icon= "file" },
                }
            });
            listdata.Add(new
            {
                id = "03",
                text = "Documents",
                icon = "folder",
                child = new List<object>() {
                    new { id= "03-01", text= "Environment Pollution.docx",
icon= "file" },
                    new { id= "03-02", text= "Global Water, Sanitation, &
Hygiene.docx", icon= "file" },
                    new { id= "03-03", text= "Global Warming.ppt", icon=
"file" },
                    new { id= "03-04", text= "Social Network.pdf", icon=
"file" },
                    new { id= "03-05", text= "Youth Empowerment.pdf", icon=
"file" }
                }
            });
        }
    }
}

```

```

listdata.Add(new
{
    id = "04",
    text = "Pictures",
    icon = "folder",
    child = new List<object>() {
        new {
            id= "04-01", text= "Camera Roll", icon= "folder",
            child= new List<object>() {
                new { id= "04-01-01", text=
"WIN_20160726_094117.JPG", icon= "file" },
                new { id= "04-01-02", text=
"WIN_20160726_094118.JPG", icon= "file" },
                new { id= "04-01-03", text=
"WIN_20160726_094119.JPG", icon= "file" }
            }
        },
        new {
            id= "04-02", text= "Wind.jpg", icon= "file"
        },
        new {
            id= "04-02", text= "Stone.jpg", icon= "file"
        },
        new {
            id= "04-02", text= "Home.jpg", icon= "file"
        },
        new {
            id= "04-02", text= "Bridge.png", icon= "file"
        }
    }
});
listdata.Add(new
{
    id = "05",
    text = "Downloads",
    icon = "folder",
    child = new List<object>() {
        new { id= "05-01", text= "UI-Guide.pdf", icon= "file" },
        new { id= "05-02", text= "Tutorials.zip", icon= "file" },

        new { id= "05-03", text= "Game.exe", icon= "file" },
        new { id= "05-04", text= "TypeScript.7z", icon= "file" },
    }
});
ViewBag.dataSource = listdata;
return View();
}
}

```

Show Items Count in Group Header

The ListView component supports wrapping list items into a group based on the category. The category of each list item can be mapped with `groupBy` field of the data source. You can display grouped list items count in the list-header using the group header template. Refer to the following code sample to display grouped list item count.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
@{
    var template = "<div class='settings'>"
        + "<div id='postContainer'><div id='postImg'>"
        + "<img src=${image} style='height:35px;width:35px;border-radius:50%;border: 1px solid #ccc;' /></div>"
        + "<div id='content'><div class='name'>${Name}</div><div id='info'>${contact}</div></div></div>";
}
@Html.EJS().ListView("List").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { GroupBy = "category", Text = "Name"
}).GroupTemplate("<div><span class='category'>${items[0].category}</span><span class='count'> ${items.length} Item(s)</span></div>").Template(template).Render()
<style>
    .count{
        float:right;
    }
    #List {
        display: block;
        margin: auto;
        border: 1px solid;
        border-color: #ccc;
        border-color: rgba(0, 0, 0, 0.12);
        width: 350px;
    }
    #List .settings {
        height: auto;
    }
    #List .e-list-group-item {
        height: 56px;
        line-height: 56px;
    }
    #List .e-list-item {
        height: 70px;
        padding: 0;
        cursor: pointer;
        box-sizing: border-box;
    }
    #List .e-list-header .e-text {
        font-family: sans-serif;
        font-size: 18px;
        line-height: 26px;
    }
    #List #content {
        margin: 9px 0 0 15px;
    }
    #List .name {
        font-size: 14px;
        line-height: 25px;
        font-weight: 500;
    }
    #info {
        line-height: 20px;
    }

```

```

        font-size: 12px;
    }
    #postImg {
        margin: 15px 9px 9px 9px;
    }
    #postContainer {
        width: inherit;
        margin: auto;
        display: inline-flex;
    }
</style>
<style>
    #text {
        margin-left: 10px;
        margin-top: 20px;
    }
    #element {
        max-width: 350px;
        margin: auto;
        margin-top: 10px;
        display: block;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new { Name = "Nancy", contact = "(206) 555-985774",
id = "1", image = "https://ej2.syncfusion.com/demos/src/grid/images/1.png",
category = "Experience" });
            listdata.Add(new { Name = "Janet", contact = "(206) 555-3412",
id = "2", image = "https://ej2.syncfusion.com/demos/src/grid/images/3.png",
category = "Fresher" });
            listdata.Add(new { Name = "Margaret", contact = "(206) 555-
8122", id = "4", image =
"https://ej2.syncfusion.com/demos/src/grid/images/4.png", category =
"Experience" });
            listdata.Add(new { Name = "Andrew ", contact = "(206) 555-9482",
id = "5", image = "https://ej2.syncfusion.com/demos/src/grid/images/2.png",
category = "Experience" });
            listdata.Add(new { Name = "Steven", contact = "(71) 555-4848",
id = "6", image = "https://ej2.syncfusion.com/demos/src/grid/images/5.png",
category = "Fresher" });

```

```

        listdata.Add(new { Name = "Michael", contact = "(71) 555-7773",
id = "7", image = "https://ej2.syncfusion.com/demos/src/grid/images/6.png",
category = "Experience" });
        listdata.Add(new { Name = "Robert", contact = "(71) 555-5598",
id = "8", image = "https://ej2.syncfusion.com/demos/src/grid/images/7.png",
category = "Fresher" });
        listdata.Add(new { Name = "Laura", contact = "(206) 555-1189",
id = "9", image = "https://ej2.syncfusion.com/demos/src/grid/images/8.png",
category = "Experience" });
        ViewBag.dataSource = listdata;
        return View();
    }
}

```

Filter List Items in the ListView

The filtered data can be displayed in the ListView component depending upon on user inputs using the [DataManager](#). Refer to the following steps to render the ListView with filtered data.

- Render a textbox to get input for filtering data.
- Render ListView with [dataSource](#), and set the [sortOrder](#) property.
- Bind the `keyup` event for textbox to perform filtering operation. To filter list data, pass the list data source to the `DataManager`, manipulate the data using the [executeLocal](#) method, and then update filtered data as ListView dataSource.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
<div id="sample">
    <input class="e-input" type="text" id="textbox" placeholder="Filter"
title="Type in a name">

@Html.EJS().ListView("list").DataSource((IEnumerable<object>)ViewBag.dataSou
rce).Fields(new ListViewFieldSettings { Id = "id", Text = "text"
}).SortOrder(SortOrder.Ascending).Render()
</div>
<style>
    #list {
        box-shadow: 0 1px 4px #ddd;
        border-bottom: 1px solid #ddd;
    }
    #sample {
        height: 220px;
        margin: 0 auto;
        display: block;
        max-width: 350px;
    }
</style>
<script>
    document.getElementById("textbox").addEventListener("keyup", onKeyUp);
    //Here, the list items are filtered using the DataManager instance for
    ListView
    function onKeyUp() {

```

```

        var listData =
@ (Html.Raw (Newtonsoft.Json.JsonConvert.SerializeObject (ViewBag.dataSource)))
;

        var listObj = (document.getElementById("list")).ej2_instances[0];
        var value = (document.getElementById("textbox")).value;
        var data = new ej.data.DataManager(listData).executeLocal(
            new ej.data.Query().where("text", "startswith", value, true)
        );
        if (!value) {
            listObj.dataSource = listData.slice();
        } else {
            listObj.dataSource = data;
        }
        listObj.dataBind();
    }
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new
            {
                text = "Hennessey Venom",
                id = "list-01"
            }); listdata.Add(new
            {
                text = "Bugatti Chiron",
                id = "list-02"
            }); listdata.Add(new
            {
                text = "Bugatti Veyron Super Sport",
                id = "list-03"
            }); listdata.Add(new
            {
                text = "SSC Ultimate Aero",
                id = "list-04"
            }); listdata.Add(new
            {
                text = "Koenigsegg CCR",
                id = "list-05"
            }); listdata.Add(new
            {
                text = "McLaren F1",
                id = "list-06"
            });
        }
    }
}

```

```

        ViewBag.dataSource = listdata;
        return View();
    }
}
}

```

Note: In this demo, data has been filtered with starting character of the list items. You can also filter list items with ending character by passing the `endswith` in [where](#) clause instead of `startswith`.

Add and Remove List Items from the ListView

You can add or remove list items from the ListView component using the [addItem](#) and [removeItem](#) methods.

Refer to the following steps to add or remove a list item.

- Render the ListView with data source, and use the [template](#) property to append the delete icon

for each list item. Also, bind the click event for the delete icon using the [actionComplete](#) handler.

- Render the Add Item button, and bind the click event. On the click event handler, pass data with random id to the [addItem](#) method to add a new list item on clicking the Add Item button.
- Bind the click handler to the delete icon created in step 1. Within the click event, remove the list item by passing the delete icon list item to [removeItem](#) method.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
<div id="sample">
    @Html.EJS().ListView("sample-list-flat").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new
    ListViewFieldSettings { IconCss = "icon", Text = "text"
    }).ActionComplete("onComplete").Template("<div class='text-content'> ${text}
    <span class = 'delete-icon'></span> </div>").Render()
    @Html.EJS().Button("btn").Content("Add Item").Render()
</div>
<style>
    #sample-list-flat {
        margin: 40px auto;
        max-width: 350px;
    }
    #btn {
        margin: 40px auto;
        display: block;
    }
    /* csslint ignore:start */
    @@font-face {
        font-family: "e-icon";
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfIAAAEoAAAVmNtYXDnEOdVAAABiAAAADZnbHlmXOn
iGAAAACgAAAFaAgVhZBC1AhkAAADQAAANmhoZWEIUQQDAAARAAAACRobXR4CAAAAAAAYAAAA
IbG9jYQCgAAAAAHAAAAABmlheHABDgCYAAABCAAAACBuYW11v4Bt4QAAAwgAAAIzCg9zdJx8QW4
AAAUkAAAAOwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAgABAAAAQAAPWcDV18
PPPUACwQAAAAAANbRXpQAAAAA1tFelAAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAACAIwAAgAAAA

```

```

AAgAAAAoACgAAAP8AAAAAAAAAAQQA ZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQM AAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAAQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
AAAACAAAAAwAAABQA AwABAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAAEAAAA
AAAAAoAAAAAIAAAAAA/QD9AALAI sAAAEHFwcnByc3JzcXNWUfHz8fLx8PHgLUhIRrg4NrhIRrg4P
9iQECAwQGBwCJCwsMDQ4PDxEREhMUFBUWFhcXFxkYGRkaGhkZGBkXFxcWFhUUFBMSERE PDw4NDAs
LCQcHBgQDAgEBAgMEBgCHCQsLDA0ODw8RERITFBQVFhYXFxcZGBkZGhoZGRgZFxcXFhYVFBQTEhE
RDw8ODQwLCwkHBwYEAwICg4OGa4SEa4ODaoCE7hoZGRgZFxcXFhYVFBQTEhERDw8ODQwLCwkHBwY
EAwIBA QIDBAYHBwKLCwWNDg8PERESExQUFRYWFxcXGRgZGRoaGRkYGRcXFxYWFQRUExIREQ8PDg0
MCwsJBwCGBAMCAQECawQGBwCJCwsMDQ4PDxEREhMUFBUWFhcXFxkYGRkAAAA SAN4AAQAAAAAAAA
BAAAAAQAAAAAAQAGAAEAQA AAAAAAgAHAACAAQAAAAAAAwAGAA4AAQAAAAAABAAGABQA AAAAA
ABQALABoAAQAAAAAABgAGACUAAQAAAAAACgAsACsAAQAAAAAACwASAFcAAwABBakAAAACAGkAAwA
BBakAAQAMAGsAAwABBakAAgAOAHcAAwABBakAAwAMAIUAwABBakABAAMAJEAAwABBakABQAWAJ0
AAwABBakABgAMALMAAwABBakACgBYAL8AAwABBakACwAkARcgZGVsZXRLUmVndWxhcmRlbGV0ZWR
lbGV0ZVZlc2lnb24gMS4wZGVsZXRLRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Zlc2lvbiBNZXR
ybyBTdHVKaW93d3cuc3luY2Zlc2lvbi5jb20AIABkAGUAbABlAHQAZQBSAGUAZwB1AGwAYQByAGQ
AZQBsAGUAdABlAGQAZQBsAGUAdABlAFYAZQByAHMAaQBvAG4AIAAxAC4AMABkAGUAbABlAHQAZQB
GAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQAIAB1AHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGk
AbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwB
uAC4AYwBvAG0AAAAAAgAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAACAQIBAwARY2lyY2x
lLWNsb3NlLS0tMDIAAAA=) format("true type");
    font-weight: normal;
    font-style: normal;
}
/* csslint ignore:end */
#sample-list-flat .delete-icon::after {
    font-family: "e-icon";
    content: "\e700";
    float: right;
    cursor: pointer;
}
</style>
<script>
    //Event handler to add the list item on button click
    document.getElementById("btn").onclick = () => {
        var listViewInstance = document.getElementById("sample-list-
flat").ej2_instances[0];
        var data = {
            text: "Koenigsegg - " + (Math.random() * 1000).toFixed(0),
            id: (Math.random() * 1000).toFixed(0).toString(),
            icon: "delete-icon"
        };
        listViewInstance.addItem([data]);
        setTimeout(() => {
            var newEle = document.querySelectorAll('[data-uid="' + data.id +
            '"]');
            newEle[0].addEventListener("click", deleteItem.bind(this));
        }, 100);
    };
    //Method for actionComplete handler
    function onComplete() {
        var iconEle = document.getElementsByClassName("delete-icon");
        //Event handler to bind the click event for delete icon
        Array.prototype.forEach.call(iconEle, (element) => {
            element.addEventListener("click", deleteItem.bind(this));
        });
    }
    //Method to delete the list item

```

```
function deleteItem(args) {
    var listViewInstance = document.getElementById("sample-list-flat").ej2_instances[0];
    args.stopPropagation();
    var liItem = (args.target).parentElement.parentElement;
    listViewInstance.removeItem(liItem);
    onComplete();
}
</script>
```

LIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new { text= "Hennessey Venom", id= "1", icon= "delete-icon" });
            listdata.Add(new { text= "Bugatti Chiron", id= "2", icon= "delete-icon" });
            listdata.Add(new { text= "Bugatti Veyron Super Sport", id= "3", icon= "delete-icon" });
            listdata.Add(new { text= "Aston Martin One- 77", id= "4", icon= "delete-icon" });
            listdata.Add(new { text= "Jaguar XJ220", id= "list-5", icon= "delete-icon" });
            listdata.Add(new { text= "McLaren P1", id= "6", icon= "delete-icon" });
            ViewBag.dataSource = listdata;
            return View();
        }
    }
}
```

Trace all events in ListView

The ListView component triggers events based on its actions. The events can be used as extension points to perform custom operations. Refer to the following steps to trace the ListView events:

1. Render the ListView with [dataSource](#), and bind the [actionBegin](#), [actionComplete](#), and [select](#) events.
2. Perform custom operations in actionBegin, actionComplete, and select events.
3. Provide event log details for actionBegin and actionComplete events, and they will be displayed in the event trace panel when the ListView action starts and the dataSource bound successfully.
4. Get the selected item details from the [SelectEventArgs](#) in the select event, and display the selected list item text in the event trace panel while selecting list items.

5. Use clear button to remove event trace information.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
<h4 id="evt-text">
    <b>Event Trace</b>
</h4>
<div id="list-container">
    <!-- ListView element -->
    @Html.EJS().ListView("listview-
def").DataSource((IEnumerable<object>)ViewBag.dataSource).ActionComplete("on
ActionComplete").ActionBegin("onActionBegin").Select("onSelect").Width("250"
).Render()
    <div id="list_event">
        <div id="evt">
            <div class="eventarea" style="height:273px;overflow: auto">
                <!-- Event log element -->
                <span class="EventLog" id="EventLog" style="word-break:
normal"></span>
            </div>
            <div class="evtbtn">
                <!-- clear button element -->
                @Html.EJS().Button("clear").Content("Clear").Render()
            </div>
        </div>
    </div>
</div>
<style>
    #list-container {
        max-width: 600px;
        margin: auto;
    }
    #EventLog b {
        color: #388e3c;
    }
    #listview-def {
        display: inline-block;
        border: 1px solid #dcdcdc;
    }
    .evtbtn {
        margin-top: 40px;
        margin-left: 70px;
    }
    #evt {
        border: 1px solid #dcdcdc;
        padding: 10px;
        width: 250px;
    }
    #list_event {
        padding-left: 40px;
        display: inline-block;
        vertical-align: top;
    }
</style>
```



```

<script>
    //Clears the event log details
    document.getElementById("clear").onclick = () => {
        document.getElementById("EventLog").innerHTML = "";
    };
    //Handler for actionBegin event trace
    function onActionBegin() {
        appendElement("<b>actionBegin </b> event is triggered<hr>");
    }
    //Handler for select event trace
    function onSelect(args) {
        appendElement(args.text + "<b>&#160;&#160;is selected</b><hr>");
    }
    //Handler for actionComplete event trace
    function onActionComplete() {
        appendElement("<b>actionComplete</b> is triggered <hr>");
    }
    //Display event log
    function appendElement(html) {
        var span = document.createElement("span");
        span.innerHTML = html;
        var log = document.getElementById("EventLog");
        log.insertBefore(span, log.firstChild);
    }
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new { text= "Hennessey Venom", id= "1", icon=
"delete-icon" });
            listdata.Add(new { text= "Bugatti Chiron", id= "2", icon=
"delete-icon" });
            listdata.Add(new { text= "Bugatti Veyron Super Sport", id= "3",
icon= "delete-icon" });
            listdata.Add(new { text= "Aston Martin One- 77", id= "4", icon=
"delete-icon" });
            listdata.Add(new { text= "Jaguar XJ220", id= "list-5", icon=
"delete-icon" });
            listdata.Add(new { text= "McLaren P1", id= "6", icon= "delete-
icon" });
            ViewBag.dataSource = listdata;
            return View();
        }
    }
}

```

```
}

```

Create Mobile contact layout from ListView

You can customize the ListView using the [template](#) property. Refer to the following steps to customize ListView as mobile contact view with our `ej2-avatar`.

- Render the ListView with [dataSource](#) that has avatar data. You can set avatar data as either text or class names. Refer to the following codes.

```
`typescript

```

```
List<object> listdata = new List<object>();

```

```
listdata.Add(new

```

```
{

```

```
text = "Jenifer",

```

```
contact = "(206) 555-985774",

```

```
id = "1",

```

```
avatar = "",

```

```
pic = "pic01"

```

```
});

```

```
listdata.Add(new

```

```
{

```

```
text = "Amenda",

```

```
contact = "(206) 555-3412",

```

```
id = "2",

```

```
avatar = "A",

```

```
pic = ""

```

```
});

```

```
`

```

- Set `avatar` classes in ListView template to customize contact icon. In the following codes, medium size avatar has been set using the class name `e-avatar e-avatar-circle` from data source.

```
`typescript

```

```
var template: "<div class='settings'>" +

```

```
"${if(avatar!=="")}" +

```

```
"<span class='e-avatar e-avatar-circle'>${avatar}</span>" +

```

```

"${else}" +
"<span class='${pic} e-avatar e-avatar-circle'> </span>" +
"${/if}" +
"<div id='content'>" +
"<div class='name'>${text}</div>" +
"<div id='info'>${contact}</div>" +
"</div>";
`

```

Note: Avatars can be set in different sizes in avatar classes. To know more about avatar classes, refer to [Avatar](#).

- Sort the contact names using the [sortOrder](#) property of ListView.
- Enable the [showHeader](#) property, and set the [headerTitle](#) as `Contacts`.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
@{
    var template = "<div class='settings'>" +
        "${if (avatar != '')}" +
        "<span class='e-avatar e-avatar-circle'>${avatar}</span>" +
        "${else}" +
        "<span class='${pic} e-avatar e-avatar-circle'> </span>" +
        "${/if}" +
        "<div id='content'>" +
        "<div class='name'>${text}</div>" +
        "<div id='info'>${contact}</div>" +
        "</div>";
}
<!-- ListView element -->
@Html.EJS().ListView("List").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { Text = "Name" })
.SortOrder(SortOrder.Ascending).Width("350").Template(template).ShowHeader(true).HeaderTitle("Contacts").Render()
<style>
    #List {
        margin: 0 auto;
        border: 1px solid #ccc;
    }
    #List .e-list-item {
        height: 60px;
        cursor: pointer;
    }
    #List .e-list-header .e-text {
        font-family: sans-serif;
        font-size: 18px;
        line-height: 16px;
    }

```

```

    }
    #List #content {
        margin: 0;
    }
    #List .e-list-header {
        background: rgb(2, 120, 215);
        color: white;
    }
    #List #info,
    #List .name {
        font-size: 14px;
        margin: 0 60px;
        line-height: 20px;
    }
    #List .name {
        padding-top: 8px;
        font-weight: 500;
    }
    .pic01 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/1.png");
    }
    .pic02 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/3.png");
    }
    .pic03 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/5.png");
    }
    .pic04 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/2.png");
    }
    #List .e-avatar {
        position: absolute;
        margin-top: 8px;
        font-size: 14px;
    }
    #List .e-list-item:nth-child(1) .e-avatar {
        background-color: #039be5;
    }
    #List .e-list-item:nth-child(2) .e-avatar {
        background-color: #e91e63;
    }
    #List .e-list-item:nth-child(6) .e-avatar {
        background-color: #009688;
    }
    #List .e-list-item:nth-child(8) .e-avatar {
        background-color: #0088;
    }
}
</style>

```

LIST.CS

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new
            {
                text= "Jenifer",
                contact= "(206) 555-985774",
                id= "1",
                avatar= "",
                pic= "pic01"
            });
            listdata.Add(new
            {
                text= "Amenda",
                contact= "(206) 555-3412",
                id= "2",
                avatar= "A",
                pic= ""
            });
            listdata.Add(new
            {
                text= "Isabella",
                contact= "(206) 555-8122",
                id= "4",
                avatar= "",
                pic= "pic02"
            });
            listdata.Add(new
            {
                text= "William ",
                contact= "(206) 555-9482",
                id= "5",
                avatar= "W",
                pic= ""
            });
            listdata.Add(new
            {
                text= "Jacob",
                contact= "(71) 555-4848",
                id= "6",
                avatar= "",
                pic= "pic04"
            });
            listdata.Add(new
            {
                text= "Matthew",
                contact= "(71) 555-7773",
                id= "7",
                avatar= "M",
```

```

        pic= ""
    });
    listdata.Add(new
    {
        text= "Oliver",
        contact= "(71) 555-5598",
        id= "8",
        avatar= "",
        pic= "pic03"
    });
    listdata.Add(new
    {
        text= "Charlotte",
        contact= "(206) 555-1189",
        id= "9",
        avatar= "C",
        pic= ""
    });
    ViewBag.dataSource = listdata;
    return View();
}
}
}

```

Display spinner until List Items are loaded

The features of the ListView component such as remote data-binding take more time to fetch data from corresponding dataSource/remote URL. In this case, you can use EJ2

[Spinner](#) to enhance the appearance of the UI. This section explains how to load a spinner component to groom the appearance.

Refer to the following code sample to render the spinner component.

```

`typescript
ej.popups.createSpinner({
target: document.getElementById('spinner')
});
ej.popups.showSpinner(document.getElementById('spinner'));
`

```

Refer to the following code sample to render the ListView component.

```

`typescript
<!-- ListView element -->
@Html.EJS().ListView("element").DataSource(dataManger =>
{dataManger.Url("https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/").CrossDomain(true);
}).Query("new
ej.data.Query().from('Products').select('ProductID,ProductName').take(10)").ActionBegin("onBegin").Fields(
new ListViewFieldSettings { Id = "ProductID", Text = "ProductName"

```

```
}).ShowHeader(true).HeaderTitle("Product
Name").Width("300").ActionComplete("oncomplete").Render()
```

Here, the data is fetched from **Northwind** Service URL; it takes a few seconds to load the data. To enhance the UI, the spinner component has been rendered initially. After the data is loaded from remote URL, the spinner component will be hidden in ListView [actionComplete](#) event.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
<!-- ListView element -->
@Html.EJS().ListView("element").DataSource(dataManger =>
{dataManger.Url("http://js.syncfusion.com/ejServices/Wcf/Northwind.svc/").Cr
ossDomain(true);
}).Query("new
ej.data.Query().from('Products').select('ProductID,ProductName').take(10)").
ActionBegin("onBegin").Fields(new ListViewFieldSettings { Id = "ProductID",
Text = "ProductName" }).ShowHeader(true).HeaderTitle("Product
Name").Width("300").ActionComplete("oncomplete").Render()
<style>
    #element {
        display: block;
        max-width: 350px;
        min-height: 200px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
</style>
<script>
    function oncomplete() {
        document.getElementById('spinner').style.display = "none";
    }
    function onBegin() {
        var ele = document.createElement('div');
        ele.id = 'spinner';
        document.getElementById('element').appendChild(ele);
        ej.popups.createSpinner({
            target: document.getElementById('spinner')
        });
        ej.popups.showSpinner(document.getElementById('spinner'));
    }
</script>
```

LIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
```

```

    {
        public IActionResult list()
        {
            return View();
        }
    }
}

```

Hide checkbox in ListView

The checkbox of the any list item can be hidden by using [htmlAttributes](#) of [fields](#) object. With the help of [htmlAttributes](#) we can add unique class to each list item that will be rendered from the data source, from the CSS class we can hide the checkbox of the list item.

In this sample, we had hidden the multiple leaf node of nested list. The `e-checkbox-hidden` class has been added in the data source where the checkbox needs to be hidden. Refer the below snippet for simple data source.

```

`typescript
{
    text= 'New York',
    id= '3002',
    category= 'USA',
    htmlAttributes= { 'class': 'e-file e-checkbox-hidden' }
}
,

```

Even though we have hidden the checkbox the functionality will be same for the list item which might affect the `getSelectedItems` method. So, to counteract that we will follow certain logic in the `select` event. The Logic here is to remove the `e-active` class from the other checkbox hidden list item which will be added when we select on that item and retain `e-active` on currently selected item.

Note: In this process we will exclude the visible checkbox list items and only consider the hidden checkbox items.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
<!-- ListView element -->
@Html.EJS().ListView("folderCheckbox").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { Tooltip = "text"
}).ShowHeader(true).HeaderTitle("Mixed Leaf Checkbox Hidden
List").ShowCheckBox(true).Select("onSelect").Render()
<style>
    #folderCheckbox {
        border: 1px solid #dddddd;
        border-radius: 3px;
        width: 350px;
        margin: auto;
    }

```



```

        #folderCheckbox .e-checkbox-hidden .e-checkbox-wrapper {
            visibility: hidden;
        }
    </style>
    <script>
        function onSelect(args) {
            var listViewInstance =
document.getElementById("folderCheckbox").ej2_instances[0];
            // Selecting all the e-active elements from the list.
            var normalElements =
Array.prototype.slice.call(listviewInstance.element.getElementsByClassName('
e-checkbox-hidden'));
            // Looping through all the selected element and removing e-active
class
            // to avoid behaviour interference with getSelectedItems method
normalElements.forEach((element) => {
                element.classList.remove('e-active');
            });
            // Finally adding e-active class to currently selected item except
checkbox item.
            // because if it is checkbox item their actions will taken care from
the source side itself.
            if (args.item.classList.contains('e-checkbox-hidden')) {
                args.item.classList.add('e-active');
            }
        }
    </script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new
            {
                text = "Asia",
                id = "01",
                category = "Continent",
                child = new List<object>() { new { text = "India", id = "1",
category = "Asia",
                child= new List<object>() {
                    new { id= "1001", text= "Delhi", category=
"India",htmlAttributes= new { @class= "e-file e-checkbox-hidden" } },
                    new {text= "Kashmir", id= "1002", category=
"India",htmlAttributes= new { @class= "e-file e-checkbox-hidden" } },
                    new { text= "Goa",id= "1003", category=
"India",htmlAttributes= new { @class= "e-file" } }

```

```

    },
    new { text = "China",id = "2",category = "Asia",
        child = new List<object>() {
            new { text = "Zhejiang", id = "2001", category =
"China",htmlAttributes= new { @class= "e-file" } },
            new {text= "Hunan",id= "2002", category=
"China",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
            new { text= "Shandong", id= "2003",category=
"China",htmlAttributes= new { @class= "e-file" }}
        }
    }
});
listdata.Add(new
{
    text = "North America",
    id = "02",
    category = "Continent",
    child = new List<object>() { new { text = "USA", id = "3",
category = "North America",
        child= new List<object>() {
            new {text= "California", id= "3001", category=
"USA",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
            new { text= "New York",id= "3002", category=
"USA",htmlAttributes= new { @class= "e-file e-checkbox-hidden" } },
            new { text= "Florida",id= "3003", category= "USA"
,htmlAttributes= new { @class= "e-file" }}
        }
    },
    new { text = "Canada",id = "4",category = "North America",
        child = new List<object>() {
            new { text = "Ontario", id = "4001", category =
"Canada",htmlAttributes= new { @class= "e-file e-checkbox-hidden" } },
            new {text= "Alberta",id= "4002", category=
"Canada",htmlAttributes= new { @class= "e-file" }},
            new { text= "Manitoba", id= "4003",category=
"Canada",htmlAttributes= new { @class= "e-file" }}
        }
    }
});
listdata.Add(new
{
    text = "Europe",
    id = "03",
    category = "Continent",
    child = new List<object>() { new { text = "Germany", id =
"5", category = "Europe",
        child= new List<object>() {
            new {text= "Berlin", id= "5001", category=
"Germany",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
            new { text= "Bavaria",id= "5002", category=
"Germany",htmlAttributes= new { @class= "e-file" } },
            new { text= "Hesse",id= "5003", category=
"Germany" ,htmlAttributes= new { @class= "e-file e-checkbox-hidden" }}
        }
    }
});

```

```

    },
    new { text = "France",id = "6",category = "Europe",
        child = new List<object>() {
            new { text = "Paris", id = "6001", category =
"France" ,htmlAttributes= new { @class= "e-file" }},
            new {text= "Lyon",id= "6002", category=
"France",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
            new { text= "Marseille", id= "6003",category=
"France",htmlAttributes= new { @class= "e-file" }}
        }
    }
});
listdata.Add(new
{
    text = "Australia",
    id = "04",
    category = "Continent",
    child = new List<object>() { new { text = "Australia", id =
"7", category = "Australia",
        child= new List<object>() {
            new {text= "Sydney", id= "7001", category=
"Australia",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
            new { text= "Melbourne",id= "7002", category=
"Australia" ,htmlAttributes= new { @class= "e-file" }},
            new { text= "Brisbane",id= "7003", category=
"Australia" ,htmlAttributes= new { @class= "e-file" }}
        }
    },
    new { text = "New Zealand",id = "8",category = "Australia",
        child = new List<object>() {
            new { text = "Milford Sound", id = "8001",
category = "New Zealand",htmlAttributes= new { @class= "e-file" } },
            new {text= "Tongariro National Park",id= "8002",
category= "New Zealand",htmlAttributes= new { @class= "e-file" }},
            new { text= "Fiordland National Park", id=
"8003",category= "New Zealand",htmlAttributes= new { @class= "e-file e-
checkbox-hidden" }}
        }
    },
});
listdata.Add(new
{
    text = "Africa",
    id = "05",
    category = "Continent",
    child = new List<object>() { new { text = "Morocco", id =
"9", category = "Africa",
        child= new List<object>() {
            new {text= "Rabat", id= "9001", category=
"Morocco",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
            new { text= "Toubkal",id= "9002", category=
"Morocco",htmlAttributes= new { @class= "e-file" } },
            new { text= "Todgha Gorge",id= "9003", category=
"Morocco" ,htmlAttributes= new { @class= "e-file e-checkbox-hidden" }}
        }
    }
});

```

```

        },
        new { text = "South Africa", id = "10", category = "Africa",
            child = new List<object>() {
                new { text = "Cape Town", id = "10001", category =
"South Africa" ,htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
                new {text= "Pretoria", id= "10002", category=
"South Africa",htmlAttributes= new { @class= "e-file e-checkbox-hidden" }},
                new { text= "Bloemfontein", id= "10003",category=
"South Africa",htmlAttributes= new { @class= "e-file" }}
            }
        },
    }
});
ViewBag.dataSource = listdata;
return View();
}
}
}

```

Manipulate ListView as Grid Layout

In Listview, list items can be rendered in grid layout with following data manipulations.

- Add Item
- Remove Item
- Sort Items
- Filter Items

Grid Layout

In this section, we will discuss about rendering of list items in grid layout.

- Initialize and render ListView with dataSource which will render list items in list layout.
- Now, add the below CSS to list item. This will make list items to render in grid layout

`css

```

element .e-list-item {
height: 100px;
width: 100px;
float: left;
}
`

```

In the below sample, we have rendered List items in grid layout.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
@{
    var template = "<img id='listImage' src='./apple.png' alt='apple' />";
}

```

```

<!-- ListView element -->
@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).Template(template).Render()
<style>
#element {
    display: block;
    max-width: 303px;
    margin: auto;
    border: 1px solid #dddddd;
    border-radius: 3px;
}
#element .e-list-item {
    height: 100px;
    width: 100px;
    float: left;
    padding: 0;
}
#listImage {
    width: 55px;
    height: 55px;
    margin-left: 20px;
    margin-top: 20px;
}
</style>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            string[] listdata = { "1", "2", "3", "4", "5", "6", "7", "8", "9"
};
            ViewBag.dataSource = listdata;
            return View();
        }
    }
}

```

Data manipulation

In this section, we will discuss about ListView data manipulations.

Add Item

We can add list item using [addItem](#) API. This will accept array of data as argument.

```
`typescript
```

```
listViewInstance.addItem([{text: 'Apricot', id: '32'}]);
```

In the below sample, you can add new fruit item by clicking add button which will open dialog box with fruit name and image URL text box. After entering the item details, click the add button. This will add your new fruit item.

Remove item

We can remove list item using [removeItem](#) API. This will accept fields with `id` or list item element as argument.

`typescript

```
listViewInstance.removeItem({id: '32'});
```

In the below sample, you can remove fruit by hovering the fruit item which will show delete button and click that delete button to delete that fruit from your list.

Sort Items

ListView can be sorted either in Ascending or Descending order. To enable sorting in your ListView, set [sortOrder](#) as `Ascending` or `Descending`.

`typescript

```
@Html.EJS().ListView("element").SortOrder(SortOrder.Ascending).Render()
```

We can also set sorting after component initialization.

`typescript

```
listViewInstance.sortOrder = 'Ascending'
```

In the below sample, we have sorted fruits in `Ascending` order. To sort it in descending, click on sort order icon and vice versa.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
@using Syncfusion.EJ2.Popups;
@{
    var template = "<div class='fruits'><div class='first'><img
id='listImage' src='${ imgUrl}' alt='fruit' /><button class='delete e-
control e-btn e-small e-round e-delete-btn e-primary e-icon-btn' data-
ripple='true'><span class='e-btn-icon e-icons delete-
icon'></span></button></div><div class='fruitName'>${text}</div></div>";
}
<div id="container">
    <div class="headerContainer">
        <div class="e-input-group">
            <input id="search" class="e-input" type="text"
placeholder="Search fruits" />
            <span class="e-input-group-icon e-input-search"></span>
        </div>
    </div>
</div>
```

```

<button id="sort" class="e-control e-btn e-small e-round e-primary
e-icon-btn" title="Sort fruits" data-ripple="true">
    <span class="e-btn-icon e-icons e-sort-icon-ascending"></span>
</button>
<button id="add" class="e-control e-btn e-small e-round e-primary e-
icon-btn" title="Add fruit" data-ripple="true">
    <span class="e-btn-icon e-icons e-add-icon"></span>
</button>

@Html.EJS().Dialog("dialog").ShowCloseIcon(true).Buttons(ViewBag.DialogButto
ns).Content("<div id='listDialog'><div class='input_name'><label
for='name'>Fruit Name: </label><input id='name' class='e-input' type='text'
placeholder='Enter fruit name' /></div><div><label for='imgurl'>Fruit Image:
</label><input id='imgurl' class='e-input' type='text' placeholder='Enter
image url' /></div></div>").Header("Add
fruit").Width("300px").Visible(false).Render()
<!-- ListView element -->

@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).ActionComplete("wireEvents").Template(template).SortOrder(SortOrder.
Ascending).Render()
</div>
</div>
<style>
    #listImage {
        width: 55px;
        height: 55px;
        margin-left: 25px;
    }
    #container {
        max-width: 440px;
        margin: auto;
        margin-top: 130px;
        box-shadow: 0 3px 6px lightgray;
    }
    #dialog {
        top:auto !important;
    }
    .headerContainer {
        height: 48px;
        line-height: 48px;
        background: rgb(2, 120, 215);
        color: white;
        margin-bottom: 3px;
    }
    .headerContainer .e-input-group {
        margin-left: 20px;
        width: 200px;
        background: white;
        height: 31px;
    }
    .headerContainer #search {
        height: 21px;
        margin-left: 10px;
    }
    #listDialog .input_name {
        margin-bottom: 20px;

```

```

}
.headerContainer #add,
.headerContainer #sort {
    float: right;
    margin-right: 15px;
    margin-top: 7px;
    background: white;
    color: black;
}
.headerContainer .e-input-search::before {
    font-family: 'e-icons';
    content: '\e961';
    margin-top: 3px;
}
.headerContainer .e-input-group .e-input-group-icon.e-input-search {
    padding: 0 10px 0 10px;
}
#element .e-list-item {
    height: 110px;
    width: 110px;
    float: left;
    padding: 0;
    position: relative;
    user-select: none;
}
#element .e-delete-btn {
    float: right;
    visibility: hidden;
    margin-top: -10px;
}
#element .e-delete-btn.e-btn.e-small.e-round {
    width: 2em;
    height: 2em;
}
#element .e-btn.e-small.e-round .e-btn-icon.delete-icon {
    font-size: 9px;
}
#element .e-list-item:hover .e-delete-btn {
    visibility: visible;
    background: red;
    border-radius: 50%;
}
#element .fruits {
    height: inherit;
    width: inherit;
    padding: 10px 0 10px 0;
}
#element .fruitName {
    text-align: center;
}
.headerContainer .e-add-icon::before {
    content: '\e823';
}
#element .delete-icon::before {
    content: '\e7fc';
    color: white;
}

```



```

        .headerContainer .e-sort-icon-ascending::before {
            content: '\e840';
        }
        .headerContainer .e-sort-icon-descending::before {
            content: '\e83f';
        }
    </style>
    <script>
        var fruitsdata =
@ (Html.Raw (Newtonsoft.Json.JsonConvert.SerializeObject (ViewBag.dataSource)))
;
        var listViewInstance;
        window.onload = function () {
            listViewInstance =
document.getElementById("element").ej2_instances[0];
        }
        function addItem() {
            var dialogObj = document.getElementById("dialog").ej2_instances[0];
            (document.getElementById("name")).value = "";
            (document.getElementById("imgurl")).value = "";
            dialogObj.show()
        }
        function wireEvents() {
            Array.prototype.forEach.call(document.getElementsByClassName('e-
delete-btn'), (ele) => {
                ele.addEventListener('click', onDeleteBtnClick);
            });
            document.getElementById("add").addEventListener('click', addItem);
            document.getElementById("sort").addEventListener('click',
sortItems);
            document.getElementById("search").addEventListener("keyup",
onKeyUp);
        }
        //Here we are removing list item
        function onDeleteBtnClick(e) {
            e.stopPropagation();
            var li = e.currentTarget.closest(".e-list-item");
            var data = listViewInstance.findItem(li);
            listViewInstance.removeItem(data);
            new ej.data.DataManager(fruitsdata).remove('id', { id: (data)["id"]
});
        }
        //Here we are adding list item
        function dlgButtonClick() {
            var dialogObj = document.getElementById("dialog").ej2_instances[0];
            var name = (document.getElementById("name")).value;
            var url = (document.getElementById("imgurl")).value;
            var id = Math.random() * 10000;
            listViewInstance.addItem([{ text: name, id: id, imgUrl: url }]);
            fruitsdata.push({ text: name, id: id, imgUrl: url });
            listViewInstance.element.querySelector('[data-uid="' + id +
""]').getElementsByClassName('e-delete-btn')[0].addEventListener('click',
onDeleteBtnClick);
            dialogObj.hide();
        }
        //Here we are sorting list item
        function sortItems() {

```

```

var ele = document.getElementById("sort").firstElementChild;
var des = ele.classList.contains('e-sort-icon-descending') ? true :
false;
if (des) {
    ele.classList.remove('e-sort-icon-descending');
    ele.classList.add('e-sort-icon-ascending');
    listViewInstance.sortOrder = 'Ascending'
} else {
    ele.classList.remove('e-sort-icon-ascending');
    ele.classList.add('e-sort-icon-descending');
    listViewInstance.sortOrder = 'Descending'
}
listViewInstance.dataBind();
wireEvents();
}
//Here, the list items are filtered using the DataManager instance.
function onKeyUp() {
    var value = (document.getElementById("search")).value;
    var data = new ej.data.DataManager(fruitsdata).executeLocal(
        new ej.data.Query().where("text", "startswith", value, true)
    );
    if (!value) {
        listViewInstance.dataSource = fruitsdata.slice();
    } else {
        listViewInstance.dataSource = data;
        listViewInstance.dataBind();
    }
}
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Popups;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<DialogDialogButton> button = new
List<DialogDialogButton>();
            List<object> listdata = new List<object>();
            listdata.Add(new { text = "Date", id = "1", imgUrl =
"./dates.jpg" });
            listdata.Add(new { text = "Fig", id = "2", imgUrl = "./fig.jpg"
});
            listdata.Add(new { text = "Apple", id = "3", imgUrl =
"./apple.png" });
            listdata.Add(new { text = "Apricot", id = "4", imgUrl =
"./apricot.jpg" });

```

```

        listdata.Add(new { text = "Grape", id = "5", imgUrl =
"./grape.jpg" });
        listdata.Add(new { text = "Strawberry", id = "6", imgUrl =
"./strawberry.jpg" });
        listdata.Add(new { text = "Pineapple", id = "7", imgUrl =
"./pineapple.jpg" });
        listdata.Add(new { text = "Melon", id = "8", imgUrl =
"./melon.jpg" });
        listdata.Add(new { text = "Lemon", id = "9", imgUrl =
"./lemon.jpg" });
        listdata.Add(new { text = "Cherry", id = "10", imgUrl =
"./cherry.jpg" });
        ViewBag.dataSource = listdata;
        button.Add(new DialogDialogButton() { Click = "dlgButtonClick",
ButtonModel = new ButtonModel() { isPrimary = true, content = "Add" } });
        ViewBag.DialogButtons = button;
        return View();
    }
    public class ButtonModel
    {
        public string content { get; set; }
        public bool isPrimary { get; set; }
    }
}

```

Drag and Drop List Items

In ListView component, we don't have drag and drop support. But we can achieve this requirement using [TreeView](#) component with ListView appearance.

Drag and Drop in TreeView component was enabled by setting [allowDragAndDrop](#) to `true`.

`typescript

```

@Html.EJS().TreeView("element").Fields(ViewBag.TreeViewFields).AllowDragAndDrop(true).Render()
,

```

The TreeView component is used to represent hierarchical data in a tree like structure. So, list items in TreeView can be dropped to child of target element. we can prevent this behaviour by cancelling the [nodeDragStop](#) and [nodeDragging](#) events.

`typescript

```

@Html.EJS().TreeView("element").Fields(ViewBag.TreeViewFields).AllowDragAndDrop(true).NodeDraggi
ng("onDragStop").NodeDragStop("onDragStop").Render()

```

```

function onDragStop(args) {

```

```

//Block the Child Drop operation in TreeView

```

```

var draggingItem = document.getElementsByClassName("e-drop-in");

```

```

if (draggingItem.length == 1) {

```

```

    draggingItem[0].classList.add('e-no-drop');

```

```

    args.cancel = true;
}

```

```

}
}
,

```

In the below sample, we have rendered draggable list items.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
using Syncfusion.EJ2.Navigations;
@Html.EJS().TreeView("element").Fields(ViewBag.TreeViewFields).AllowDragAndDrop(true).NodeDragging("onDragStop").NodeDragStop("onDragStop").Render()
<style>
    #element.e-treeview .e-ul {
        padding: 0;
    }
    #element.e-treeview .e-list-item {
        padding: 0 16px;
    }
    #element.e-treeview .e-text-content {
        padding: 0;
    }
    #element.e-treeview .e-fullrow {
        height: 36px;
    }
    #element.e-treeview .e-list-text {
        line-height: 34px;
    }
    #element.e-treeview .e-list-item:last-child {
        margin-bottom: 9px;
    }
    #element.e-treeview .e-list-item:first-child {
        margin-top: 9px;
    }
</style>
<script>
    function onDragStop(args) {
        //Block the Child Drop operation in TreeView
        var draggingItem = document.getElementsByClassName("e-drop-in");
        if (draggingItem.length == 1) {
            draggingItem[0].classList.add('e-no-drop');
            args.cancel = true;
        }
    }
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Navigations;
namespace WebApplication1.Controllers

```

```
{  
    public class ListViewController : Controller  
    {  
        public IActionResult list()  
        {  
            List<object> listdata = new List<object>();  
            listdata.Add(new  
            {  
                text = "Hennessey Venom",  
                id = "list-01"  
            }); listdata.Add(new  
            {  
                text = "Bugatti Chiron",  
                id = "list-02"  
            }); listdata.Add(new  
            {  
                text = "Bugatti Veyron Super Sport",  
                id = "list-03"  
            }); listdata.Add(new  
            {  
                text = "SSC Ultimate Aero",  
                id = "list-04"  
            }); listdata.Add(new  
            {  
                text = "Koenigsegg CCR",  
                id = "list-05"  
            }); listdata.Add(new  
            {  
                text = "McLaren F1",  
                id = "list-06"  
            }); listdata.Add(new  
            {  
                text = "Aston Martin One- 77",  
                id = "list-07"  
            }); listdata.Add(new  
            {  
                text = "Jaguar XJ220",  
                id = "list-08"  
            }); listdata.Add(new  
            {  
                text = "McLaren P1",  
                id = "list-09"  
            }); listdata.Add(new  
            {  
                text = "Ferrari LaFerrari",  
                id = "list-10"  
            });  
            ViewBag.dataSource = listdata;  
            return View();  
        }  
    }  
}
```

Render ListView with hyper-link navigation

We can use `anchor` tag along with `href` attribute in our ListView `template` property for navigation.

```
`typescript
```

```
var anchor_template = "<a target='blank' href='${url}'>${name}</a>";
```

```
,
```

In the below sample, we have rendered **ListView** with search engines URL.

CSHTML

```
@{
    var anchor_template = "<a target='_blank' href='${url}'>${name}</a>";
}
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
@Html.EJS().ListView("element").DataSource((IEnumerable<object>)ViewBag.data
Source).Template(anchor_template).HeaderTitle("Search
engines").ShowHeader(true).Render()
<style>
    #element {
        display: block;
        max-width: 350px;
        margin: auto;
        border: 1px solid #dddddd;
        border-radius: 3px;
    }
    #element a {
        text-decoration: none;
    }
    #element .e-list-header {
        background: rgb(2, 120, 215);
        color: white;
        font-size: 19px;
        font-weight: 500;
    }
</style>
```

LIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listdata = new List<object>();
            listdata.Add(new { name = "Google", url =
"https://www.google.com" });
            listdata.Add(new { name = "Bing", url = "https://www.bing.com"
});
            listdata.Add(new { name = "Yahoo", url = "https://www.yahoo.com"
});
        }
    }
}
```

```

        listdata.Add(new { name = "Ask.com", url = "https://www.ask.com"
    });
        listdata.Add(new { name = "AOL.com", url =
    "https://www.aol.com"});
        ViewBag.dataSource = listdata;
        return View();
    }
}

```

Customize ListView as Chat Window

ListView can be customized as chat window. To achieve that, use the ListView [template](#) property and [Avatar](#) component.

- The Listview template property is used to showcase the ListView as chat window.
- Avatar component is used to design the image of contact person.

Refer the below template code snippet for Template of chat window.

```

`typescript
var template = "<div class='settings'>" +
    "${if(chat!=='receiver')} " +
    "<div id='content'>" +
    "<div class='name'>${text}</div>" +
    "<div id='info'>${contact}</div></div>" +
    "${if(avatar!=='')} " +
    "<div id='image'><span class='e-avatar img1 e-avatar-circle'>${avatar}</span></div>" +
    "${else}" +
    "<div id='image'><span class='${pic} img1 e-avatar e-avatar-circle'> </span></div>" +
    "${/if}" +
    "${else}" +
    "${if(avatar!=='')} " +
    "<div id='image2'><span class='e-avatar img2 e-avatar-circle'>${avatar}</span></div>" +
    "${else}" +
    "<div id='image2'><span class='${pic} img2 e-avatar e-avatar-circle'> </span></div>" +
    "${/if}" +
    "<div id='content1'>" +
    "<div class='name1'>${text}</div>" +
    "<div id='info1'>${contact}</div>" +
    "</div>" +

```

```
"${/if}" +
"</div>";
`
```

Chat order in template

In ListView template, we have rendered the list items based on receiver and sender information from dataSource of listview.

Adding messages to chat window

- Use textbox to get message from user.
- Add the textbox message to ListView dataSource using [addItem](#) method.

```
`typescript
```

```
document.getElementById('btn').addEventListener('click', (e) => {
var value = document.getElementById('name').value;

document.getElementById('List').ej2_instances[0].addItem([{ text: "Amenda", contact: value, id: "2",
avatar: "A", pic: "", chat: "receiver" }]);

});
`
```

CSHTML

```
@{
    var template = "<div class='settings'>" +
        "${if(chat!=='receiver')}" +
        "<div id='content'>" +
        "<div class='name'>${text}</div>" +
        "<div id='info'>${contact}</div></div>" +
        "${if(avatar!=='')}" +
        "<div id='image'><span class='e-avatar img1 e-avatar-circle'>${avatar}</span></div>" +
        "${else}" +
        "<div id='image'><span class='${pic} img1 e-avatar e-avatar-circle'> </span></div>" +
        "${/if}" +
        "${else}" +
        "${if(avatar!=='')}" +
        "<div id='image2'><span class='e-avatar img2 e-avatar-circle'>${avatar}</span></div>" +
        "${else}" +
        "<div id='image2'><span class='${pic} img2 e-avatar e-avatar-circle'> </span></div>" +
        "${/if}" +
        "<div id='content1'>" +
        "<div class='name1'>${text}</div>" +
        "<div id='info1'>${contact}</div>" +
        "</div>" +
        "${/if}" +
        "</div>";
}
```



```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists;
@Html.EJS().ListView("List").DataSource((IEnumerable<object>)ViewBag.dataSource).Fields(new ListViewFieldSettings { Text = "text"
}).Template(template).HeaderTitle("Chat").Width("350px").ShowHeader(true).Render()
<div style="width: 350px;margin: 0 auto;">
<input id="name" style="width: 275px" class="e-input" type="text"
placeholder="Type your message" />
@Html.EJS().Button("btn").Content("Send").Render()
</div>
<style>
    #btn {
        float: right;
    }
    #List {
        margin: 0 auto;
        border: 1px solid #ccc;
    }
    #List .e-list-item {
        height: auto;
        cursor: pointer;
    }
    #List .e-list-header .e-text {
        font-family: sans-serif;
        font-size: 18px;
        line-height: 26px;
    }
    #List #info,
    #List .name {
        font-size: 11px;
        line-height: 20px;
    }
    #List .name {
        padding-top: 3px;
        font-weight: 500;
        padding-left: 150px;
    }
    #List #info {
        float: right;
        margin-right: 10px;
    }
    .pic01 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/1.png");
    }
    .pic02 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/3.png");
    }
    .pic03 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/5.png");
    }
    .pic04 {
        background-image:
url("https://ej2.syncfusion.com/demos/src/grid/images/2.png");

```

```

    }
    .img2.e-avatar {
        margin-left: 10px;
        margin-top: 2px !important;
        font-size: 13px;
    }
    #List #content1 {
        width: 200px;
        background-color: aliceblue;
        display: inline-block;
        margin: 5px;
    }
    #List #info1,
    #List .name1 {
        font-size: 11px;
        line-height: 20px;
        margin-left: 10px;
    }
    #List .name1 {
        padding-top: 3px;
        font-weight: 500;
    }
    #List #content {
        margin: 5px;
        width: 200px;
        margin-left: 90px;
        background-color: aliceblue;
        display: inline-block;
    }
    #image {
        float: right;
        display: inline-block;
    }
    #image2 {
        float: left;
        display: inline-block;
    }
    .img1.e-avatar {
        margin-right: 10px;
        margin: 5px;
        font-size: 13px;
    }
    .e-listview .e-list-item {
        padding: 0px !important;
    }
    .e-listview .e-list-header {
        color: white !important;
    }
    .e-listview .e-list-header {
        background: rgb(2, 120, 215) !important;
    }
    #List.e-listview .e-list-item.e-hover {
        background-color: transparent;
    }
</style>
<script>
    document.getElementById('btn').addEventListener('click', (e) => {

```

```

        let value = document.getElementById('name').value;
        document.getElementById('List').ej2_instances[0].addItem([{ text:
"Amenda", contact: value, id: "2", avatar: "A", pic: "", chat: "receiver"
}]);
        document.getElementById('name').value = "";
    });
</script>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> listData = new List<object>();
            listData.Add(new
            {
                text = "Jenifer",
                contact = "Hi",
                id = "1",
                avatar = "",
                pic = "pic01",
                chat = "sender"
            });
            listData.Add(new { text = "Amenda", contact = "Hello", id = "2",
avatar = "A", pic = "", chat = "receiver" });
            listData.Add(new
            {
                text = "Jenifer",
                contact = "What Knid of application going to launch",
                id = "4",
                avatar = "",
                pic = "pic02",
                chat = "sender"
            });
            listData.Add(new
            {
                text = "Amenda ",
                contact = "A knid of Emergency broadcast App",
                id = "5",
                avatar = "A",
                pic = "",
                chat = "receiver"
            });
            listData.Add(new
            {
                text = "Jacob",
                contact = "Can you please elaborate",
                id = "6",
            });
        }
    }
}

```

```

        avatar = "",
        pic = "pic04",
        chat = "sender"
    });
    ViewBag.dataSource = listData;
    return View();
}
}
}

```

Customize Each List Item's Text in ListView Component

To customize the text of each list item in the ListView component, you can make use of the [HtmlAttributes](#) property. This property allows you to specify HTML attributes and styles for individual items within the list, enabling you to create visually appealing and interactive lists tailored to your requirements.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Lists
<div>

@Html.EJS().ListView("listview").Enable(true).CssClass("custom").DataSource(
    (IEnumerable<object>)ViewBag.dataSource).Render()
</div>
<style>
    .custom.e-listview .e-list-item.e-high .e-text-content {
        color: red;
    }
    .custom.e-listview .e-list-item.e-moderate .e-text-content {
        color: yellow;
    }
    .custom.e-listview .e-list-item.e-normal .e-text-content {
        color: black;
    }
</style>

```

LIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
    public class ListViewController : Controller
    {
        public IActionResult list()
        {
            List<object> data = new List<object>();
            data.Add(new { text = "Hennessey Venom", id = "list-01",
            htmlAttributes = new Dictionary<string, object>() { { "class", "e-high" } }
            });
        }
    }
}

```

```

        data.Add(new { text = "Bugatti Chiron", id = "list-02",
htmlAttributes = new Dictionary<string, object>() { { "class", "e-moderate"
} } });
        data.Add(new { text = "Bugatti Veyron Super Sport", id = "list-
03", htmlAttributes = new Dictionary<string, object>() { { "class", "e-
normal" } } });
        data.Add(new { text = "SSC Ultimate Aero", id = "list-04",
htmlAttributes = new Dictionary<string, object>() { { "class", "e-moderate"
} } });
        data.Add(new { text = "Koenigsegg CCR", id = "list-05",
htmlAttributes = new Dictionary<string, object>() { { "class", "e-normal" }
} });
        data.Add(new { text = "McLaren F1", id = "list-06",
htmlAttributes = new Dictionary<string, object>() { { "class", "e-high" } }
});
        data.Add(new { text = "Aston Martin One- 77", id = "list-07",
htmlAttributes = new Dictionary<string, object>() { { "class", "e-moderate"
} } });
        ViewBag.dataSource = data;
        return View();
    }
}
}

```

Maps

Getting Started with ASP.NET MVC Maps Component

This section briefly explains about how to include [ASP.NET MVC MapsLink to the Video](#) component in your ASP.NET MVC application using Visual Studio.

You can explore some useful features in the Maps component using the following video.

Prerequisites

[System requirements for ASP.NET MVC components](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Maps component

Now, add the Syncfusion ASP.NET MVC Maps component in **~/Views/Home/Index.cshtml** page.

CSHTML

```
@Html.EJS().Maps("container").Layers(layer => {
layer.ShapeData(Model).Add(); }).Render();
```

Place the **WorldMap.json** file in **App_Data** folder of the project. Read the content in **WorldMap.json** file in the code behind and assign the deserialized object to the **shapeData** property of the Maps component using the **ViewBag** object in **HomeController.cs** as shown below.

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    return View(GetWorldMap());
}
public object GetWorldMap()
{
    string allText =
        System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
    return JsonConvert.DeserializeObject(allText, typeof(object));
}
```

Note: The `shapeData` in the `ViewBag` object is already assigned to the Maps component.

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Maps component will be rendered in the default web browser.



Note: [View Sample in GitHub.](#)

Populate data

Geometry types

GeoJSON data contains geometry objects with properties such as geometry types and coordinates. The geometry types are the values present in the geometry objects of the GeoJSON data that specify the type of shape to be rendered, as well as the coordinates that help to draw the shape's boundary line. The supportive geometry types are:

| Shapes | Supported |

| --- | --- |

| Polygon | Yes |

| MultiPolygon | Yes |

| LineString | Yes |

| MultiLineString | Yes |

| Point | Yes |

| MultiPoint | Yes |

| GeometryCollection | Yes |

[Shape data](#)

The shape data collection describes geographical shape information that is available in GeoJSON format. The Map shapes are rendered with this data. The custom shapes such as seat selection in bus, seat selection in a cricket stadium and more useful information can be also added as ShapeData in the layer of the Maps.

[Data source](#)

The **DataSource** property is used to represent statistical data in the Maps component, and it accepts a collection of values as input. For example, a list of objects as input can be provided to the data source. This data source will be used to color the map, display data labels, and display tooltip, among other things.

The data source is populated with JSON data relative to shape data and stored as JSON object. In the below example, the below JSON object can be used as data source in Maps.

```
`json
[
{
'code': 'AF',
'value': 53,
'name': 'Afghanistan',
'population': 29863010,
'density': 119
},
{
'code': 'AL',
'value': 117,
'name': 'Albania',
'population': 3195000,
'density': 111
```



```
,  
{  
  'code': 'DZ',  
  'value': 15,  
  'name': 'Algeria',  
  'population': 34895000,  
  'density': 15  
},  
{  
  'code': 'AO',  
  'value': 15,  
  'name': 'Angola',  
  'population': 18498000,  
  'density': 15  
},  
{  
  'code': 'AR',  
  'value': 15,  
  'name': 'Argentina',  
  'population': 40091359,  
  'density': 14  
},  
{  
  'code': 'AM',  
  'value': 109,  
  'name': 'Armenia',  
  'population': 3230100,  
  'density': 108  
}  
]  
,
```

Data binding

The following properties in the `Layers` are used for binding data in the Maps component. Both the properties are related to each other.

- ShapePropertyPath
- ShapeDataPath

ShapePropertyPath

The **ShapePropertyPath** property is used to refer the field name in the **ShapeData** property of shape layers to identify the shape. When the values of **ShapeDataPath** property from the **DataSource** property and **ShapePropertyPath** property from the **ShapeData** property match, then the associated object from the data source is bound to the corresponding shape.

Note: **world-map.json** file contains following data and its field **name** value is used to map the corresponding shape with the provided data source.

```
`json
{
  "type": "Feature",
  "properties": {
    "admin": "Afghanistan",
    "name": "Afghanistan",
    "continent": "Asia"
  },
  "geometry": { "type": "Polygon", "coordinates": [[[61.21081709172573, ... ],
...
}
`
```

ShapeDataPath

The **ShapeDataPath** property is similar to the **ShapePropertyPath** property, but it refers to the field name in the **DataSource** property. For example, [populationData](#) contains the **code**, **value**, **name**, **population** and **density** fields. Here, the **name** field is set to the **shapeDataPath** to map the corresponding value of field name in shape data.

In the below example, both **name** fields contain the same value as **Afghanistan**, this value is matched in both shape data and data source, so that the details associated with **Afghanistan** will be mapped to the corresponding shape and used to color the corresponding shape, display data labels, display tooltips, and more.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Height("450px").Width("650px").Layers(m =>
{
    m.ShapeSettings(ss =>
ss.Fill("#E5E5E5").ColorValuePath("color")).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").DataSource(ViewBag.populationData).Add();
}).Render()
```

POPULATE-DATA.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetPopulationData()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/populationdensity.js"
);
            return JsonConvert.DeserializeObject(text);
        }
    }
}
```



Binding complex data source

Data from data source can be bind to the Maps in two different ways.

1. Bind the field name directly to the properties as `ShapeDataPath`, `ColorValuePath`, `ValuePath` and `ShapeValuePath`.
2. Bind the field name as `data.field` to the properties as `ShapeDataPath`, `ColorValuePath`, `ValuePath` and `ShapeValuePath`.

Refer the data values for [ViewBag.bubbleData](#), [ViewBag.complexData](#) and [ViewBag.markerData](#) here.

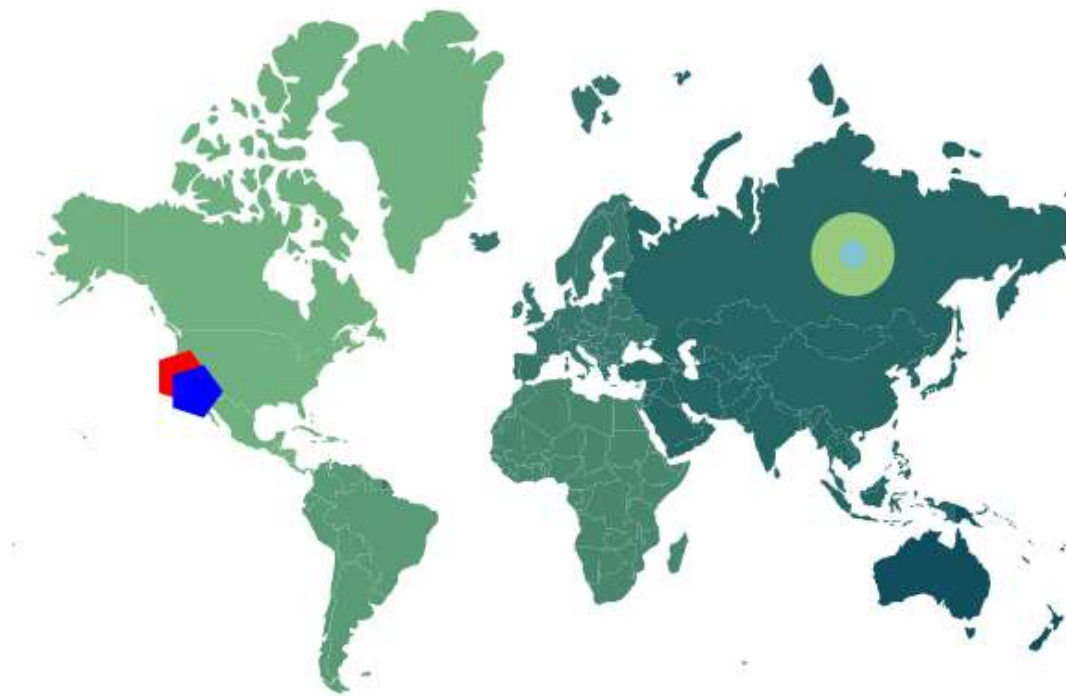
CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Height("450px").Width("650px").Layers(m =>
{
    m.ShapeData(ViewBag.worldmap).ShapeDataPath("data.name").ShapePropertyPath("name").DataSource(ViewBag.populationData).Add();
}).Render()
```

COMPLEX-DATA.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.bubbleData = GetBubbleData();
            ViewBag.markerData = GetMarkerData();
            ViewBag.complexData = GetComplexData();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetComplexData()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/ComplexData.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetBubbleData()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/BubbleData.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMarkerData()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/MarkerData.json");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



Layers

The Maps component is rendered through [Layers](#) and any number of layers can be added to the Maps.

Multilayer

The Multilayer support allows loading multiple shape files and map providers in a single container, enabling Maps to display more information. The shape layer or map providers are the main layers of the Maps. Multiple layers can be added as **SubLayer** over the main layers using the [Type](#) property of [Layers](#).

Sublayer

Sublayer is a type of shape file layer. It allows loading multiple shape files in a single map view. For example, a sublayer can be added over the main layer to view geographic features such as rivers, valleys and cities in a map of a country. Similar to the main layer, elements in the Maps such as markers, bubbles, color mapping and legends can be added to the sub-layer.

In this example, the United States map shape is used as shape data by utilizing **usa.ts** file, and **texas.ts** and **california.ts** files are used as sub-layers in the United States map.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss.Fill("#E5E5E5").Border(border =>
border.Color("black").Width(0.1).Opacity(1)))
    .ShapeData(ViewBag.usamap).Add();
    l.ShapeSettings(ss => ss.Fill("rgba(141, 206, 255, 0.6)").Border(border
=> border.Color("#1a9cff").Width(0.25).Opacity(1)))
```

```
.ShapeData(ViewBag.texasmap).Type(Syncfusion.EJ2.Maps.Type.SubLayer).Add();
    l.ShapeSettings(ss => ss.Fill("rgba(141, 206, 255, 0.6)").Border(border
=> border.Color("#1a9cff").Width(0.25).Opacity(1)))

.ShapeData(ViewBag.californiamap).Type(Syncfusion.EJ2.Maps.Type.SubLayer).Ad
d();
}).Render()
```

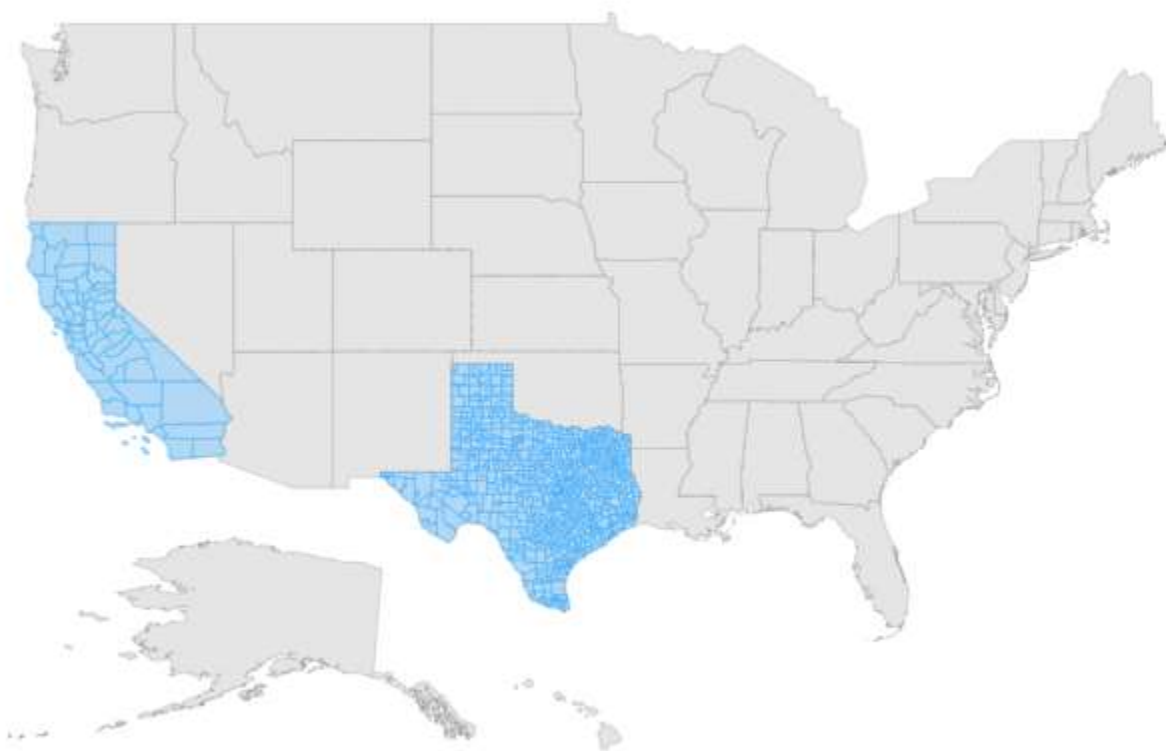
LAYERS.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.california = GetCaliforniaMap();
            ViewBag.texas = GetTexasMap();
            ViewBag.usamap = GetUSMaps();
            ViewBag.californiamap = GetCaliforniaMaps();
            ViewBag.texasmap = GetTexasMaps();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/USA.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetCaliforniaMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/California.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetTexasMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/Texas.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUSMaps()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
```

```

        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetCaliforniaMaps()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/California.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetTexasMaps()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Texas.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



Displaying different layer in the view

Multiple shape files and map providers can be loaded simultaneously in Maps. The **BaseLayerIndex** property is used to determine which layer on the user interface should be displayed. This property is used for the Maps drill-down feature, so when the **BaseLayerIndex** value is changed, the corresponding shape is loaded. In this example, two layers can be loaded with the World map and the United States map. Based on the given **BaseLayerIndex** value the corresponding shape will be loaded in the user interface. If the **BaseLayerIndex** value is set to **0**, then the world map will be loaded.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").BaseLayerIndex(1).Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
    l.ShapeData(ViewBag.usaMap).Add();
}).Render()
```

LAYERVIEW.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.usa = GetUSMap();
            ViewBag.worldMap = GetWorldMaps();
            ViewBag.usaMap = GetUSMaps();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/USA.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetWorldMaps()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetUSMaps()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```

```
}  
}
```



Rendering custom shapes

Providers

OpenStreetMaps in ASP.NET MVC Maps Component

The OpenStreetMap (OSM) is the online Maps provider built by a community of developers; it is free to use under an open license. It allows to view geographical data in a collaborative way from anywhere on the earth. The OSM Maps provides small tile images based on our requests and combines those images into a single image to display the Maps area in the Maps component.

Adding OpenStreetMap

The OSM Maps can be rendered using the [UriTemplate](#) property.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().Maps("maps").Layers(l=> {  
  
    l.UriTemplate("http://a.tile.openstreetmap.org/level/tileX/tileY.png").Add()  
    ;  
}).Render()
```

OSM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

**Changing the tile server of the OpenStreetMap**

The OSM tile server can be changed by setting the tile URL in the `UrlTemplate` property. For more details about the OSM tile server, refer [here](#).

Enabling zooming and panning

The OSM Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

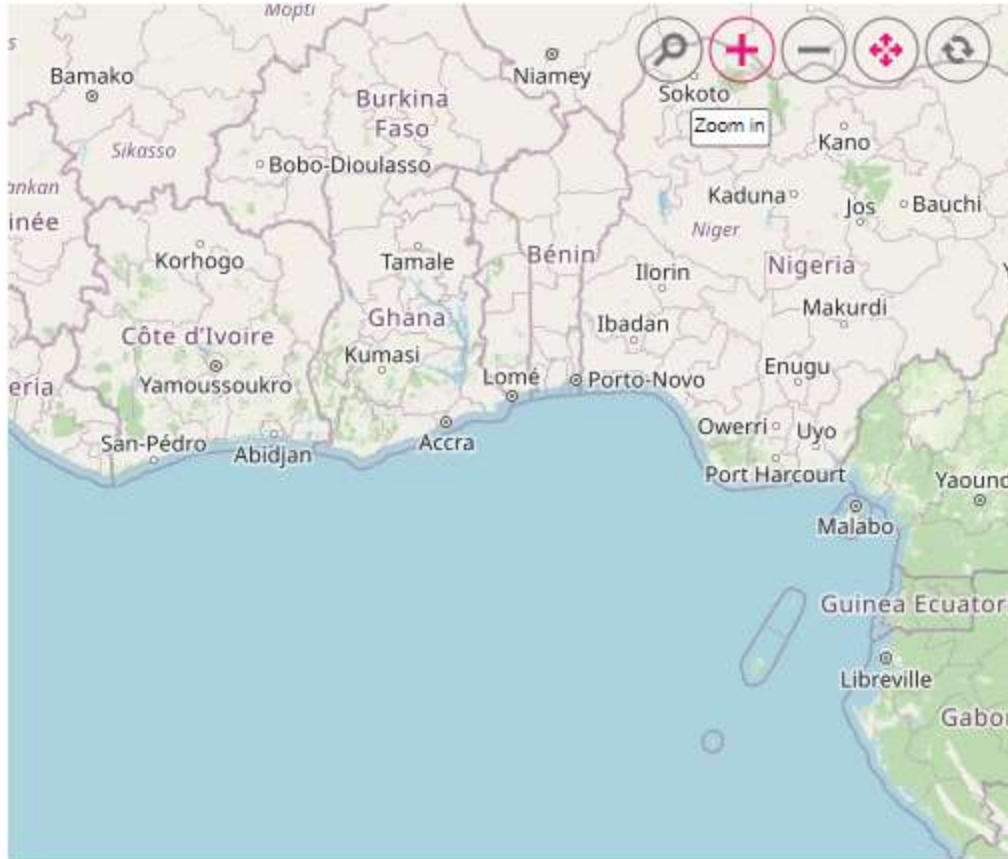
CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true)).Layers(l=> {

    l.UrlTemplate("http://a.tile.openstreetmap.org/level/tileX/tileY.png").Add()
;
}).Render()
```

OSMZOOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Adding markers and navigation line

Markers can be added to the layers of OSM Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerSettings](#). Navigation lines can be added on top of an OSM Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineSettings](#).

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.UrlTemplate("http://a.tile.openstreetmap.org/level/tileX/tileY.png").MarkerSettings(marker =>
    {
        marker.Visible(true).DataSource(ViewBag.markerData).Add();
    }).NavigationLineSettings(ns =>
    {
        ns.Visible(true).Latitude(new double[] { 34.060620, 40.724546 })
        .Longitude(new double[] { -118.330491, -73.850344
    }).Color("black").Angle(90)
        .Width(2).DashArray("4").Add();
    }).Add();
}).Render()
```

OSMMARKER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<MarkerData> data = new List<MarkerData>
            {
                new MarkerData {latitude= 37.0000, longitude= -120.0000,
city= "California" },
                new MarkerData {latitude= 40.7127, longitude= -74.0059,
city= "New York" },
                new MarkerData {latitude= 42.0000, longitude= -93.0000,
city= "Iowa" }
            };
            ViewBag.markerData = data;
            return View();
        }
        public class MarkerData
        {
            public double latitude { get; set; }
            public double longitude { get; set; }
            public string city { get; set; }
        }
    }
}
```



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the OSM Maps layer for highlighting a particular continent or country in OSM Maps by adding another layer and specifying the [Type](#) property of Maps layer to **SubLayer**.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.UrlTemplate("http://a.tile.openstreetmap.org/level/tileX/tileY.png").Add();
    l.ShapeSettings(s =>
    s.Fill("blue")).ShapeData(ViewBag.africaMapShape).Type(Syncfusion.EJ2.Maps.Type.SubLayer).Add();
}).Render()
```

SUBLAYER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.africeMap = GetAfricaMap();
            ViewBag.africaMapShape = GetAfricaShape();
            return View();
        }
        public object GetAfricaMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Africa.json");
            return JsonConvert.DeserializeObject(allText);
        }

        public object GetAfricaShape()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Africa.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```




Enabling legend

The legend can be added to the tile Maps by setting the [Visible](#) property of [LegendSettings](#) to **true**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps
@{
    var urlTemplate =
    "https://tile.openstreetmap.org/level/tileX/tileY.png";
}
<div class="control-section">
    <div id="outer" style="width:100%">

@Html.EJS().Maps("container").Load("mapsLoad").UseGroupingSeparator(true).Format("n").TitleSettings(title => title.Text("Top 10 populated cities in the World").TextStyle(new MapsFont { Size = "16px", FontFamily = "inherit", Opacity = 1 })).LegendSettings(legend =>
legend.Position(LegendPosition.Float).Visible(true).Type(LegendType.Markers).Background("#E6E6E6").Height("170px").TextStyle(new MapsFont { Color = "#000000", FontFamily = "inherit" })).Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>
    {
        new Syncfusion.EJ2.Maps.MapsLayer
        {
            AnimationDuration = 0, UrlTemplate = urlTemplate,
```

```

MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
{
    new Syncfusion.EJ2.Maps.MapsMarker
    {
        Visible = true, Height = 15, Width = 15,
        ColorValuePath = "color", LegendText = "name", Shape = MarkerType.Circle,
        AnimationDuration = 0,
        DataSource = new [] {
            new { name="Tokyo",
latitude=35.6805245924747, longitude=139.76770396213337,
population=37435191, color="#2EB6C8"},
            new { name="Delhi", latitude=28.644800,
longitude=77.216721, population=29399141, color="#4A97F4"},
            new { name="Shanghai",
latitude=31.224361, longitude=121.469170, population=26317104,
color="#498082"},
            new { name="Sao Paulo", latitude=-
23.550424484747914, longitude=-46.646471636488315, population=21846507,
color="#FB9E67"},
            new { name="Mexico City",
latitude=19.427402397418774, longitude=-99.131123716666,
population=21671908, color="#8F9DE3"},
            new { name="Cairo ", latitude=30.033333,
longitude=31.233334, population=20484965, color="#7B9FB0"},
            new { name="Dhaka", latitude=23.777176,
longitude=90.399452, population=20283552, color="#4DB647"},
            new { name="Mumbai",
latitude=19.08492049646163, longitude=72.87449446319248,
population=20185064, color="#30BEFF"},
            new { name="Beijing",
latitude=39.90395970055848, longitude=116.38831272088059,
population=20035455, color="#Ac72AD"},
            new { name="Osaka",
latitude=34.69024500601642, longitude=135.50746225677142,
population=19222665, color="#EFE23E"}
        },
        TooltipSettings = new MapsTooltipSettings{
Visible=true, ValuePath="name", Format= "City Name: ${name} <br> Population:
${population} million" },
    }
}
}).Render()

</div>
</div>
<style>
    #container {
        display: block;
    }
</style>
<script type="text/javascript">
    function mapsLoad(args) {
        args.maps.legendSettings.location.x = 10;
        args.maps.legendSettings.location.y = 262;
    }
</script>

```

LEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

**Bing Maps in ASP.NET MVC Maps Component**

Bing Maps is an online Maps provider, owned by Microsoft. As like OSM, it provides Maps tile images based on our requests and combines those images into a single one to display the Maps area.

Adding Bing Maps

The Bing Maps can be rendered using the [UrlTemplate](#) property, which is based on the URL generated by the `GetBingUrlTemplate` method in the Maps. The format of the required URL of Bing Maps varies from other online map providers. As a result, a built-in `GetBingUrlTemplate` method has been included that returns the URL in a generic format. In the meantime, a subscription key is required for Bing Maps. The Bing Maps key can be obtained from [here](#), then append it to the Bing Maps URL before passing it to the `GetBingUrlTemplate` method. The URL returned by this method must be passed to the [UrlTemplate](#) property.

CSHTML

```
@using Syncfusion.EJ2;
<div class="control-section">
    <div id="outer" style="width:100%">
        @Html.EJS().Maps("container").Load("mapsLoad").Layers(l =>
        {
            l.Add();
        }).Render()
    </div>
</div>
<style>
    #container {
        display: block;
    }
</style>
<script type="text/javascript">
    function mapsLoad(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/M
etadata/CanvasLight?output=json&uriScheme=https&key=?").then(function (url)
{
    args.maps.layers[0].urlTemplate = url;
});
    }
</script>
```

BING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

```
}
```



Types of Bing Maps

Bing Maps provides different types of Maps and it is supported in the Maps component.

- **Aerial** - Displays satellite images to highlight roads and major landmarks for easy identification.
- **AerialWithLabel** - Displays aerial Maps with labels for the continent, country, ocean, etc.
- **Road** - Displays the default Maps view of roads, buildings, and geography.
- **CanvasDark** - Displays dark version of the road Maps.
- **CanvasLight** - Displays light version of the road Maps.
- **CanvasGray** - Displays grayscale version of the road Maps.

To render the light version of the road Maps, set the **CanvasLight** value is passed via the URL into the `GetBingUrlTemplate` method demonstrated in the following code sample.

CSHTML

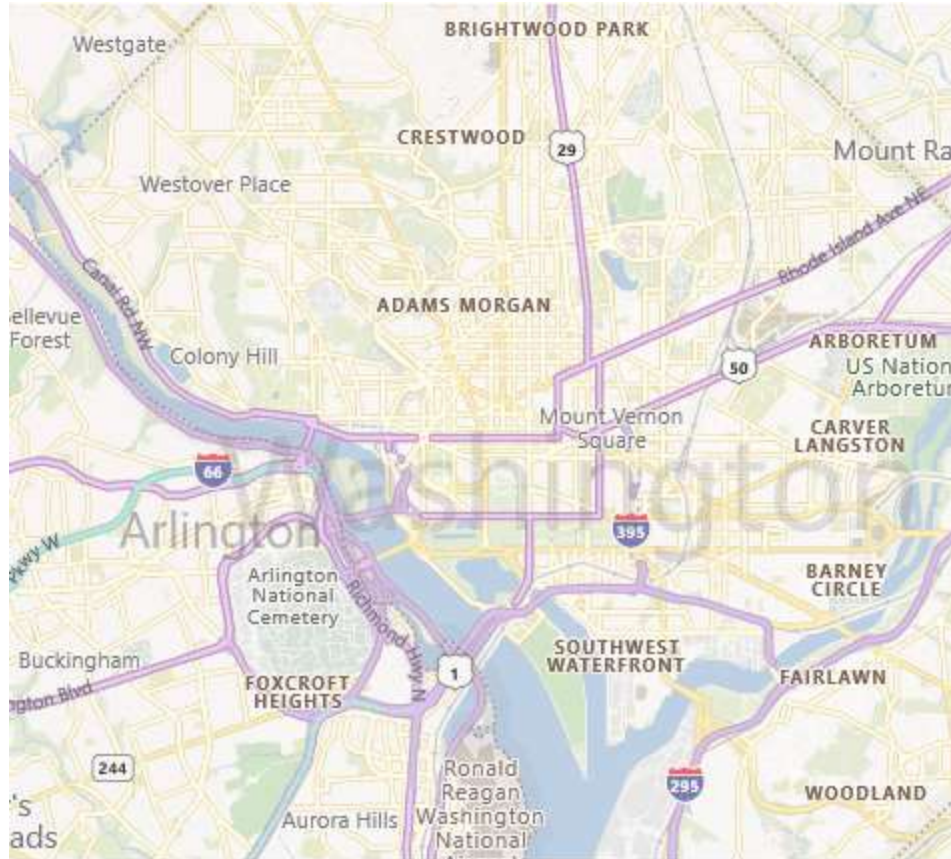
```
@using Syncfusion.EJ2;  
<div class="control-section">  
  <div id="outer" style="width:100%">  
    @Html.EJS().Maps("container").Load("mapsLoad").Layers(l =>  
    {  
      l.Add();  
    }).Render()  
  </div>  
</div>
```

```
</div>
<style>
    #container {
        display: block;
    }
</style>
<script type="text/javascript">
    function mapsLoad(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/M
etadata/CanvasLight?output=json&uriScheme=https&key=?").then(function (url)
{
    args.maps.layers[0].urlTemplate = url;
});
    }
</script>
```

BING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Enabling zooming and panning

Bing Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Load("mapsLoad").ZoomSettings(zoom=>zoom.Enable(true)).Layers(l=> {
    l.Add();
}).Render()
<script type="text/javascript">
    function mapsLoad(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/Metadata/CanvasLight?output=json&uriScheme=https&key=?").then(function (url)
{
    args.maps.layers[0].urlTemplate = url;
});
    }
</script>
```

BINGZOOM.CS

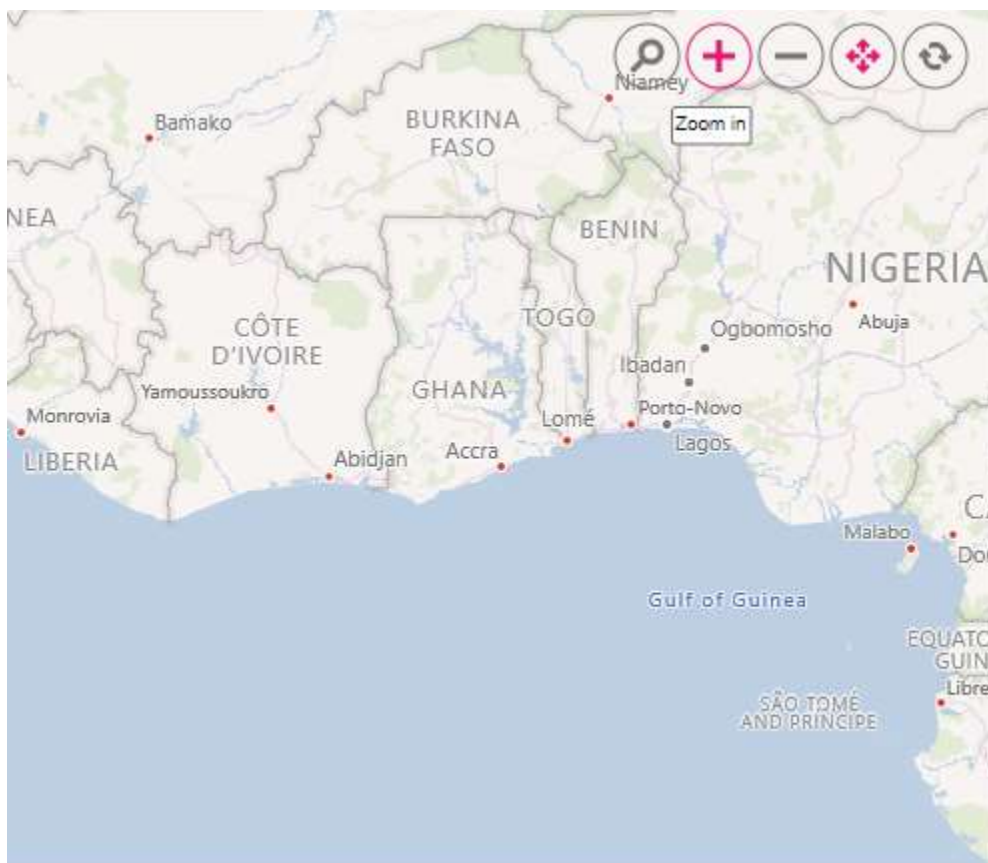
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```



```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Adding markers and navigation line

Markers can be added to the layers of Bing Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerSettings](#). Navigation lines can be added on top of an Bing Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineSettings](#).

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Load("mapsLoad").Layers(1 =>

```



```

{
    l.MarkerSettings(marker =>
    {
        marker.Visible(true).DataSource(ViewBag.markerData).Add();
    }).NavigationLineSettings(ns =>
    {
        ns.Visible(true).Latitude(new double[] { 34.060620, 40.724546 })
        .Longitude(new double[] { -118.330491, -73.850344
    }).Color("black").Angle(90)
        .Width(2).DashArray("4").Add();
    }).Add();
    }).Render()
<script type="text/javascript">
    function mapsLoad(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/M
etadata/CanvasLight?output=json&uriScheme=https&key=?").then(function (url)
{
    args.maps.layers[0].urlTemplate = url;
});
    }
</script>

```

BINGMARKER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<MarkerData> data = new List<MarkerData>
            {
                new MarkerData {latitude= 37.0000, longitude= -120.0000,
city= "California" },
                new MarkerData {latitude= 40.7127, longitude= -74.0059,
city= "New York" },
                new MarkerData {latitude= 42.0000, longitude= -93.0000,
city= "Iowa" }
            };
            ViewBag.markerData = data;
            return View();
        }
        public class MarkerData
        {
            public double latitude { get; set; }
            public double longitude { get; set; }
            public string city { get; set; }
        }
    }
}

```

```

    }
  }
}

```



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Bing Maps layer for highlighting a particular continent or country in Bing Maps by adding another layer and specifying the [Type](#) property of Maps layer to **SubLayer**.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Load("mapsLoad").Layers(l =>
{
    l.Add();
    l.ShapeSettings(s =>
s.Fill("blue")).ShapeData(ViewBag.africaShape).Type(Syncfusion.EJ2.Maps.Type
.SubLayer).Add();
}).Render()
<script type="text/javascript">
    function mapsLoad(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/M
etadata/Aerial?output=json&uriScheme=https&key=?").then(function (url) {
    args.maps.layers[0].urlTemplate = url;

```

```
    });  
}  
</script>
```

BINGSUBLAYER.CS

```
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            ViewBag.africa = GetAfricaMap();  
            ViewBag.africaShape = GetAfricaShape();  
            return View();  
        }  
        public object GetAfricaShape()  
        {  
            string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Africa.json"));  
            return JsonConvert.DeserializeObject(allText, typeof(object));  
        }  
        public object GetAfricaMap()  
        {  
            string text =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Africa.json");  
            return JsonConvert.DeserializeObject(text);  
        }  
    }  
}
```



Enabling legend

The legend can be added to the tile Maps by setting the [Visible](#) property of [LegendSettings](#) to **true**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps
<div class="control-section">
    <div id="outer" style="width:100%">

@Html.EJS().Maps("container").Load("mapsLoad").UseGroupingSeparator(true).Fo
rmat("n").TitleSettings(title => title.Text("Top 10 populated cities in the
World").TextStyle(new MapsFont { Size = "16px", FontFamily = "inherit",
Opacity = 1 })).ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings
{
    ZoomFactor = 2,
    Enable = true
}).LegendSettings(legend =>
legend.Position(LegendPosition.Float).Visible(true).Type(LegendType.Markers)
.Background("#E6E6E6").Height("170px").TextStyle(new MapsFont { Color =
"#000000", FontFamily = "inherit" })).Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        AnimationDuration = 0,
```

```

MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
{
    new Syncfusion.EJ2.Maps.MapsMarker
    {
        Visible = true, Height = 15, Width = 15,
        ColorValuePath = "color", LegendText = "name", Shape = MarkerType.Circle,
        AnimationDuration = 0,
        DataSource = new [] {
            new { name="Tokyo",
latitude=35.6805245924747, longitude=139.76770396213337,
population=37435191, color="#2EB6C8"},
            new { name="Delhi", latitude=28.644800,
longitude=77.216721, population=29399141, color="#4A97F4"},
            new { name="Shanghai",
latitude=31.224361, longitude=121.469170, population=26317104,
color="#498082"},
            new { name="Sao Paulo", latitude=-
23.550424484747914, longitude=-46.646471636488315, population=21846507,
color="#FB9E67"},
            new { name="Mexico City",
latitude=19.427402397418774, longitude=-99.131123716666,
population=21671908, color="#8F9DE3"},
            new { name="Cairo ", latitude=30.033333,
longitude=31.233334, population=20484965, color="#7B9FB0"},
            new { name="Dhaka", latitude=23.777176,
longitude=90.399452, population=20283552, color="#4DB647"},
            new { name="Mumbai",
latitude=19.08492049646163, longitude=72.87449446319248,
population=20185064, color="#30BEFF"},
            new { name="Beijing",
latitude=39.90395970055848, longitude=116.38831272088059,
population=20035455, color="#Ac72AD"},
            new { name="Osaka",
latitude=34.69024500601642, longitude=135.50746225677142,
population=19222665, color="#EFE23E"}
        },
        TooltipSettings = new MapsTooltipSettings{
Visible=true, ValuePath="name", Format= "City Name: ${name} <br> Population:
${population} million" },
    }
}
}).Render()

</div>
</div>
<style>
    #container {
        display: block;
    }
</style>
<script type="text/javascript">
    function mapsLoad(args) {

args.maps.getBingUrlTemplate("https://dev.virtualearth.net/REST/V1/Imagery/M
etadata/CanvasLight?output=json&uriScheme=https&key=?").then(function (url)
{

```

```
        args.maps.layers[0].urlTemplate = url;
    });
    args.maps.legendSettings.location.x = 10;
    args.maps.legendSettings.location.y = 262;
}
</script>
```

BINGLEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Azure Maps in ASP.NET MVC Maps Component

Azure Maps is yet another online Maps provider, owned by Microsoft. As like OSM and Bing Maps, it provides Maps tile images based on our requests and combines those images into a single one to display Maps area.

Adding Azure Maps

The Azure Maps can be rendered using the [UrlTemplate](#) property with the tile server URL provided by online map providers. In the meantime, a subscription key is required for Azure Maps. Follow the steps in this [link](#) to generate an API key, and then added the key to the URL.

Note: Refer to [Azure Maps Licensing](#).

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l=> {
    l.UrlTemplate("https://atlas.microsoft.com/map/imagery/png?subscription-
key=Your-Key &api-
version=1.0&style=satellite&zoom=level&x=tileX&y=tileY").Add();
}).Render()
<style>
    #maps {
        display: block;
    }
</style>
```

AZURE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

*Enabling zooming and panning*

The Azure Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a map for in-depth analysis. Panning helps to move a map around to focus the targeted area.

CSHTML


```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true)).Layers(l=> {
    l.UrlTemplate("https://atlas.microsoft.com/map/imagery/png?subscription-
key=Your-Key &api-
version=1.0&style=satellite&zoom=level&x=tileX&y=tileY").Add();
}).Render()
<style>
    #maps {
        display: block;
    }
</style>
```

AZUREZOOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Adding markers and navigation line

Markers can be added to the layers of Azure Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerSettings](#). Navigation lines can be added on top of the Azure Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineSettings](#).

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").ZoomSettings(zoom =>
zoom.ZoomFactor(4)).CenterPosition(c=>c.Latitude(29.394708).Longitude(-
94.954653)).Layers(l =>
{
    l.UrlTemplate("https://atlas.microsoft.com/map/imagery/png?subscription-
key=Your-Key &api-
version=1.0&style=satellite&zoom=level&x=tileX&y=tileY").MarkerSettings(mark
er =>
{
    marker.Visible(true).DataSource(ViewBag.markerData).Height(25).Width(15).Add
();
    }).NavigationLineSettings(ns =>
{
    ns.Visible(true).Latitude(new double[] { 34.060620, 40.724546 })
    .Longitude(new double[] { -118.330491, -73.850344
}).Color("Blue").Angle(0.1).Add();
    }).Add();
    }).Render()
<style>
#maps {
    display: block;
}
</style>
```

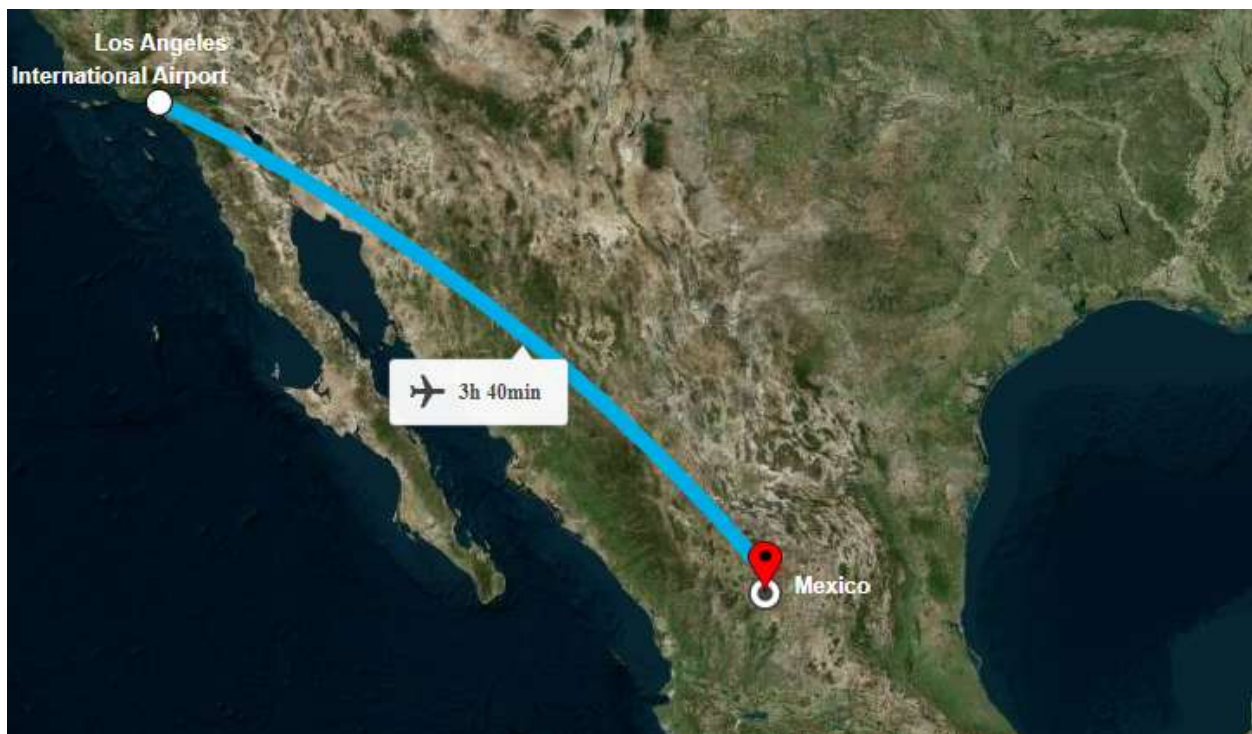
AZUREMARKER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<MarkerData> data = new List<MarkerData>
            {
                new MarkerData {latitude= 34.060620, longitude= -
118.330491, city= "California" },
            }
```

```

        new MarkerData {latitude= 40.724546, longitude= -73.850344,
city= "New York" }
        };
        ViewBag.markerData = data;
        return View();
    }
    public class MarkerData
    {
        public double latitude { get; set; }
        public double longitude { get; set; }
        public string city { get; set; }
    }
}

```



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the Azure Maps layer for highlighting a particular continent or country in Azure Maps by adding another layer and specifying the [Type](#) property of Maps layer to **SubLayer**.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.UrlTemplate("https://atlas.microsoft.com/map/imagery/png?subscription-
key=Your-Key &api-
version=1.0&style=satellite&zoom=level&x=tileX&y=tileY").Add();
}

```

```

        l.ShapeSettings(s =>
s.Fill("blue")).ShapeData(ViewBag.africaMapShape).Type(Syncfusion.EJ2.Maps.Type.SubLayer).Add();
    }).Render()
<style>
    #maps {
        display: block;
    }
</style>

```

AZURESUBLAYER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.africeMap = GetAfricaMap();
            ViewBag.africaMapShape = GetAfricaShape();
            return View();
        }
        public object GetAfricaMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Africa.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetAfricaShape()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Africa.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Enabling legend

The legend can be added to the tile Maps by setting the [Visible](#) property of [LegendSettings](#) to **true**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps
@{
    var urlTemplate =
    "https://atlas.microsoft.com/map/imagery/png?subscription-key=Your-Key &api-
    version=1.0&style=satellite&zoom=level&x=tileX&y=tileY";
}
<div class="control-section">
    <div id="outer" style="width:100%">

@Html.EJS().Maps("container").Load("mapsLoad").UseGroupingSeparator(true).Fo
rmat("n").TitleSettings(title => title.Text("Top 10 populated cities in the
World").TextStyle(new MapsFont { Size = "16px", FontFamily = "inherit",
Opacity = 1 })).LegendSettings(legend =>
legend.Position(LegendPosition.Float).Visible(true).Type(LegendType.Markers)
.Background("#E6E6E6").Height("170px").TextStyle(new MapsFont { Color =
"#000000", FontFamily = "inherit" })).Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>
    {
        new Syncfusion.EJ2.Maps.MapsLayer
        {
            AnimationDuration = 0, UrlTemplate = urlTemplate,
            MarkerSettings = new List<Syncfusion.EJ2.Maps.MapsMarker>
```

```

        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true, Height = 15, Width = 15, ColorValuePath =
"color", LegendText = "name", Shape = MarkerType.Circle, AnimationDuration =
0,

                DataSource = new [] {
                    new { name="Tokyo", latitude=35.6805245924747,
longitude=139.76770396213337, population=37435191, color="#2EB6C8"},
                    new { name="Delhi", latitude=28.644800, longitude=77.216721,
population=29399141, color="#4A97F4"},
                    new { name="Shanghai", latitude=31.224361,
longitude=121.469170, population=26317104, color="#498082"},
                    new { name="Sao Paulo", latitude=-23.550424484747914,
longitude=-46.646471636488315, population=21846507, color="#FB9E67"},
                    new { name="Mexico City", latitude=19.427402397418774,
longitude=-99.131123716666, population=21671908, color="#8F9DE3"},
                    new { name="Cairo ", latitude=30.033333,
longitude=31.233334, population=20484965, color="#7B9FB0"},
                    new { name="Dhaka", latitude=23.777176, longitude=90.399452,
population=20283552, color="#4DB647"},
                    new { name="Mumbai", latitude=19.08492049646163,
longitude=72.87449446319248, population=20185064, color="#30BEFF"},
                    new { name="Beijing", latitude=39.90395970055848,
longitude=116.38831272088059, population=20035455, color="#Ac72AD"},
                    new { name="Osaka", latitude=34.69024500601642,
longitude=135.50746225677142, population=19222665, color="#EFE23E"}
                },
                TooltipSettings = new MapsTooltipSettings{ Visible=true,
ValuePath="name", Format= "City Name: ${name} <br> Population: ${population}
million" },
            }
        }
    }).Render()

</div>
</div>
<style>
    #container {
        display: block;
    }
</style>
<script type="text/javascript">
    function mapsLoad(args) {
        args.maps.legendSettings.location.x = 10;
        args.maps.legendSettings.location.y = 262;
    }
</script>

```

AZURELEGEND.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;

```



```
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```



Other Maps providers

Apart from OpenStreetMap and Bing Maps, you can also render Maps from other online map service providers by specifying the URL provided by those providers in the [UrlTemplate](#) property. The URL template concept has been implemented in such a way that any online map service providers using the following template can benefit from previewing their Map in the Syncfusion ASP.NET MVC Maps component.

<!-- markdownlint-disable MD034 -->

Sample Template: https://< domain_name >/maps/basic/{z}/{x}/{y}.PNG

- "{z}" - It represents zoom factor (level).
- "{x}" - It indicates tile image x-position (tileX).
- "{y}" - It indicates tile image y-position (tileY).

In this case, the key generated for those online map service providers can also be appended to the URL. This allows to create personalized Maps with your own content and imagery.

Below is the example of adding TomTom map. You can follow the steps in this [link](#) to generate an API key, and then added the key to the URL.

CHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l=> {

    l.UrlTemplate("http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY
.png?key=subscription_key").Add();
}).Render()
```

OTHERMAP.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```




Enabling zooming and panning

Tile Maps layer can be zoomed and panned. Zooming helps to get a closer look at a particular area on a Maps for in-depth analysis. Panning helps to move a Maps around to focus the targeted area.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true)).Layers(l=> {
    l.UrlTemplate("http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY
.png?key=subscription_key").Add();
}).Render()
```

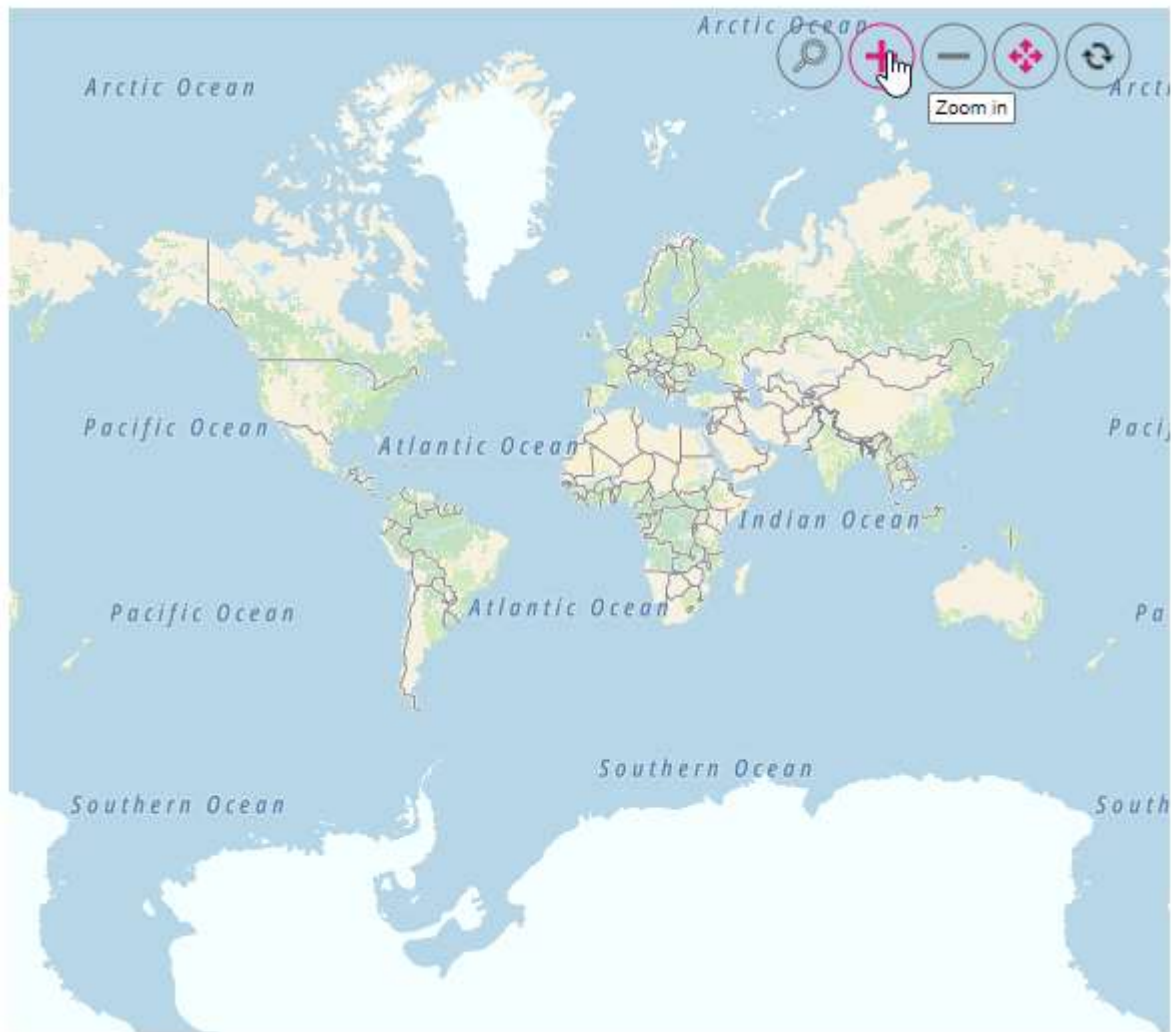
OTHERZOOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```

using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}

```



Adding markers and navigation line

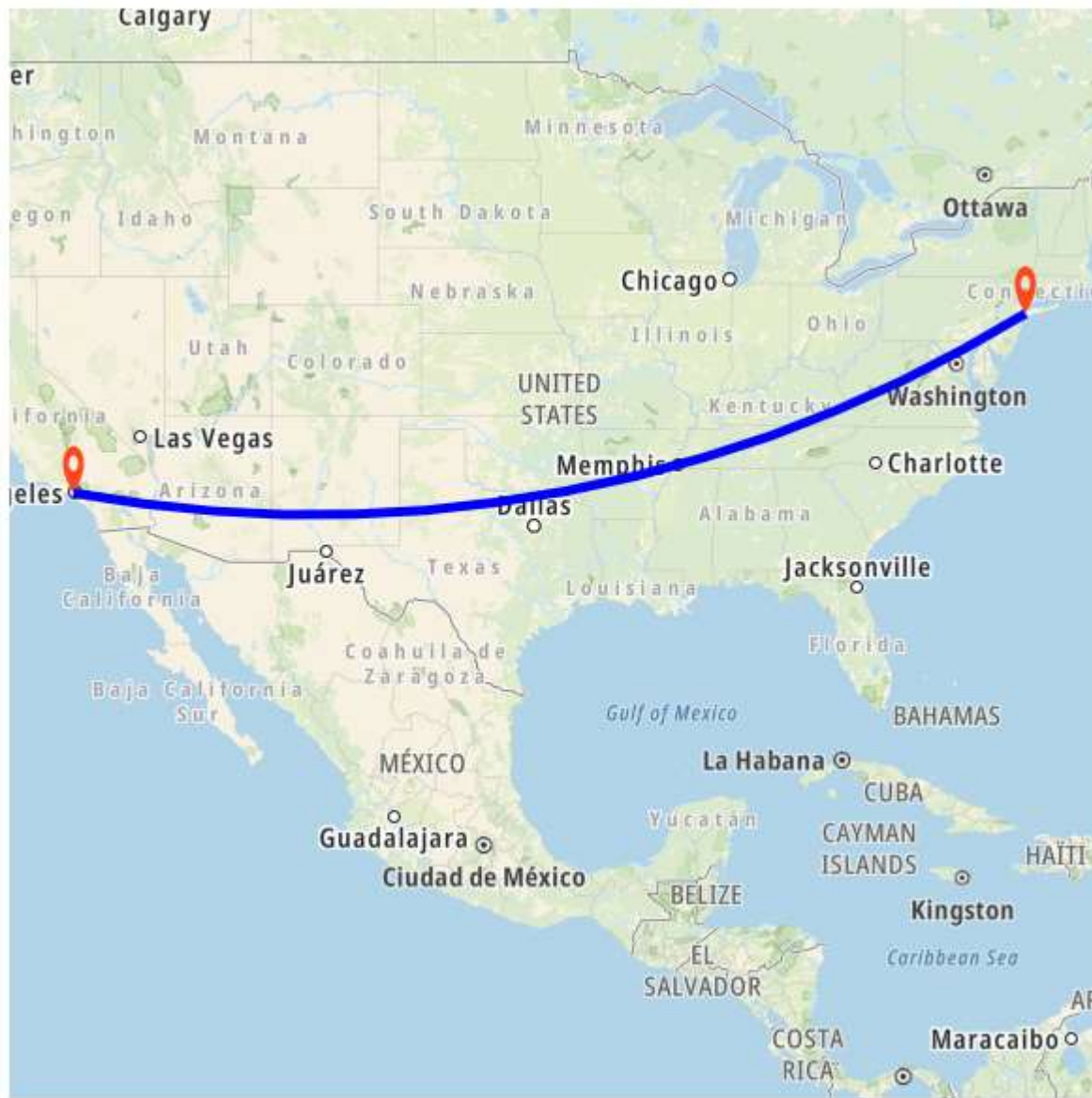
Markers can be added to the layers of tile Maps by setting the corresponding location's coordinates of latitude and longitude using [MarkerSettings](#). Navigation lines can be added on top of an tile Maps layer for highlighting a path among various places by setting the corresponding location's coordinates of latitude and longitude in the [NavigationLineSettings](#).

CHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    1.UrlTemplate("http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY
.png?key=subscription_key").MarkerSettings(marker =>
        {
            marker.Visible(true).DataSource(ViewBag.markerData).Add();
        }).NavigationLineSettings(ns =>
        {
            ns.Visible(true).Latitude(new double[] { 34.060620, 40.724546 })
            .Longitude(new double[] { -118.330491, -73.850344
        }).Color("black").Angle(90)
            .Width(2).DashArray("4").Add();
        }).Add();
    }).Render();
```

OTHERMARKER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            List<MarkerData> data = new List<MarkerData>
            {
                new MarkerData {latitude= 37.0000, longitude= -120.0000,
city= "California" },
                new MarkerData {latitude= 40.7127, longitude= -74.0059,
city= "New York" },
                new MarkerData {latitude= 42.0000, longitude= -93.0000,
city= "Iowa" }
            };
            ViewBag.markerData = data;
            return View();
        }
        public class MarkerData
        {
            public double latitude { get; set; }
            public double longitude { get; set; }
            public string city { get; set; }
        }
    }
}
```



Adding sublayer

Any GeoJSON shape can be rendered as a sublayer on top of the tile Maps layer for highlighting a particular continent or country in tile Maps by adding another layer and specifying the [Type](#) property of Maps layer to **SubLayer**.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    1.UrlTemplate("http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key").Add();
    1.ShapeData(ViewBag.africaMapShape).Type("SubLayer").Add();
}).Render()
```

OTHERSUBLAYER.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.africaMap = GetAfricaMap();
            ViewBag.africaMapShape = GetAfricaMapShape();
            return View();
        }
        public object GetAfricaMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Africa.json");
            return JsonConvert.DeserializeObject(allText);
        }

        public object GetAfricaMapShape()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Africa.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Enabling legend

The legend can be added to the tile Maps by setting the [Visible](#) property of [LegendSettings](#) to **true**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps
@{
    var urlTemplate =
    "http://api.tomtom.com/map/1/tile/basic/main/level/tileX/tileY.png?key=subscription_key";
}
<div class="control-section">
    <div id="outer" style="width:100%">

@Html.EJS().Maps("container").Load("mapsLoad").UseGroupingSeparator(true).Format("n").TitleSettings(title => title.Text("Top 10 populated cities in the World")).TextStyle(new MapsFont { Size = "16px", FontFamily = "inherit", Opacity = 1 })
        .LegendSettings(legend =>
            legend.Position(LegendPosition.Float).Visible(true).Type(LegendType.Markers)
```



```

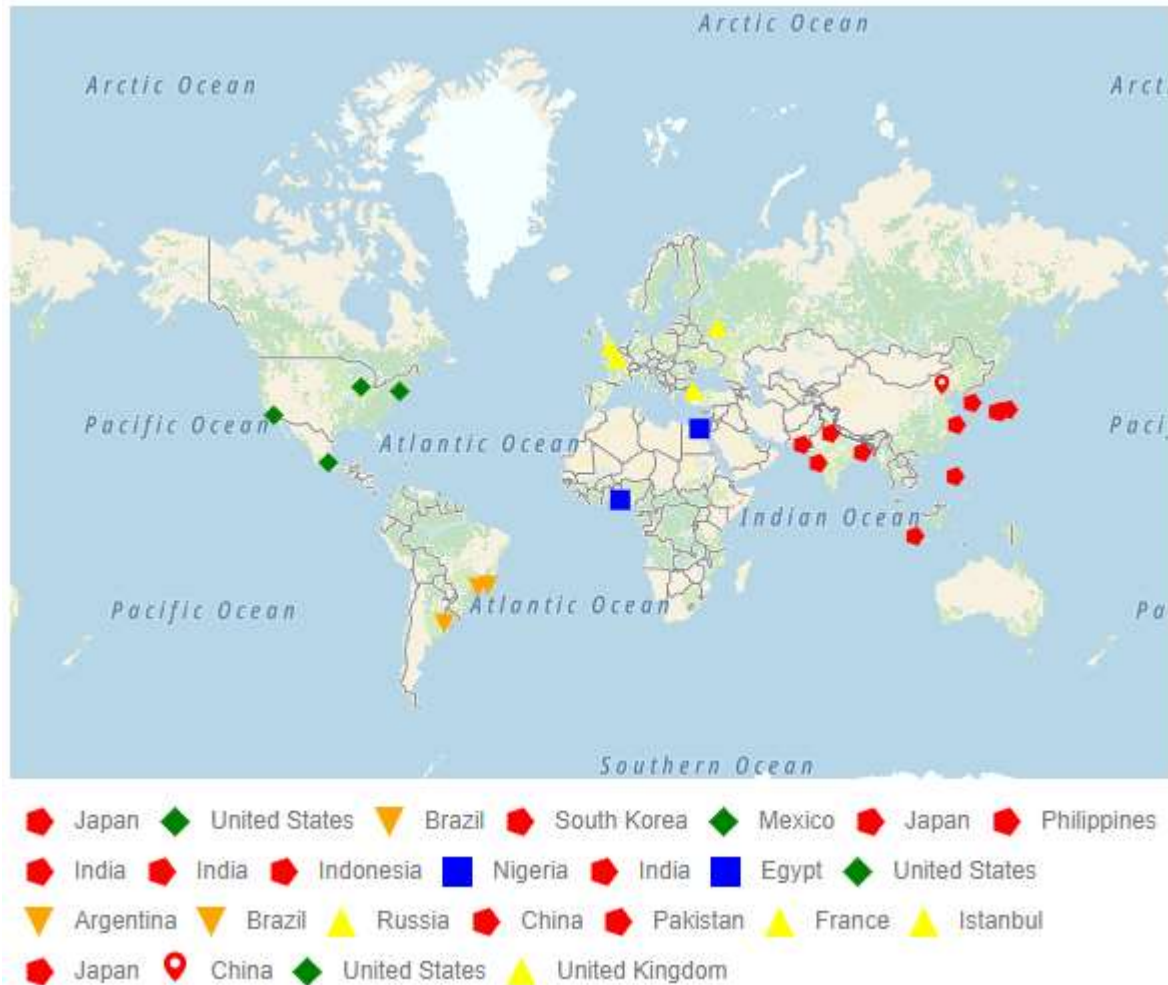
.Background("#E6E6E6").Height("170px").TextStyle(new MapsFont { Color =
"#000000", FontFamily = "inherit" })).Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        AnimationDuration = 0, UrlTemplate = urlTemplate,
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
{
    new Syncfusion.EJ2.Maps.MapsMarker
    {
        Visible = true, Height = 15, Width = 15,
        ColorValuePath = "color", LegendText = "name", Shape = MarkerType.Circle,
        AnimationDuration = 0,
        DataSource = new [] {
            new { name="Tokyo",
latitude=35.6805245924747, longitude=139.76770396213337,
population=37435191, color="#2EB6C8"},
            new { name="Delhi", latitude=28.644800,
longitude=77.216721, population=29399141, color="#4A97F4"},
            new { name="Shanghai",
latitude=31.224361, longitude=121.469170, population=26317104,
color="#498082"},
            new { name="Sao Paulo", latitude=-
23.550424484747914, longitude=-46.646471636488315, population=21846507,
color="#FB9E67"},
            new { name="Mexico City",
latitude=19.427402397418774, longitude=-99.131123716666,
population=21671908, color="#8F9DE3"},
            new { name="Cairo ", latitude=30.033333,
longitude=31.233334, population=20484965, color="#7B9FB0"},
            new { name="Dhaka", latitude=23.777176,
longitude=90.399452, population=20283552, color="#4DB647"},
            new { name="Mumbai",
latitude=19.08492049646163, longitude=72.87449446319248,
population=20185064, color="#30BEFF"},
            new { name="Beijing",
latitude=39.90395970055848, longitude=116.38831272088059,
population=20035455, color="#Ac72AD"},
            new { name="Osaka",
latitude=34.69024500601642, longitude=135.50746225677142,
population=19222665, color="#EFE23E"}
        },
        TooltipSettings = new MapsTooltipSettings{
Visible=true, ValuePath="name", Format= "City Name: ${name} <br> Population:
${population} million" },
    }
}
}).Render()
</div>
</div>
<style>
    #container {
        display: block;
    }

```

```
</style>
<script type="text/javascript">
    function mapsLoad(args) {
        args.maps.legendSettings.location.x = 10;
        args.maps.legendSettings.location.y = 262;
    }
</script>
```

OTHERLEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

Customization in ASP.NET MVC Maps Component

Setting the size for Maps

The width and height of the Maps can be set using the **Width** and **Height** properties in the Maps component. Percentage or pixel values can be used for the height and width values.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Height("500px").Width("300px").Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()
```

SIZE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Maps title

The title for the Maps can be set using the `MapsTitleSettings`. It can be customized using the following properties.

- **Alignment** - To customize the alignment for the text in the title for the Maps. The possible values are **Center**, **Near** and **Far**.
- **Description** - To set the description of the title in Maps.
- **Text** - To set the text for the title in Maps.
- **TextStyle** - To customize the text of the title in Maps.
- **SubtitleSettings** - To customize the subtitle for the Maps.

CSHTML

```
@using Syncfusion.EJ2.Maps;  
@using Syncfusion.EJ2;  
@{  
    var title = new Syncfusion.EJ2.Maps.MapsTitleSettings  
    {  
        Text = "Maps control",  
        TextStyle = new Syncfusion.EJ2.Maps.MapsFont  
        {  
            Color = "red",  
            FontFamily = "arial",  
            FontStyle = "italic",
```

```

        FontWeight = "regular",
        Size = "14px"
    },
    Alignment = Syncfusion.EJ2.Maps.Alignment.Center
};
}
@Html.EJS().Maps("maps").TitleSettings(title).Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

TITLE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

Maps control



Setting theme

The Maps component supports following themes.

- Material
- Fabric
- Bootstrap
- Highcontrast
- MaterialDark
- FabricDark
- BootstrapDark
- Bootstrap4
- HighContrastLight
- Tailwind

By default, the Maps are rendered by the **Material** theme. The theme of the Maps component is changed using the **Theme** property.

CSHTML

```
@using Syncfusion.EJ2;
```

```
@Html.EJS().Maps("maps").Theme(Syncfusion.EJ2.Maps.MapsTheme.MaterialDark).Layers(l =>
{
    l.ShapeData(ViewBag.worldmap).Add();
}).Render()
```

THEME.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Customizing Maps container

The following properties are available to customize the container in the Maps.

- **Background** - To apply the background color to the container in the Maps.
- **Border** - To customize the color, width and opacity of the border of the Maps.
- **Margin** - To customize the margins of the Maps.

CSHTML

```
@using Syncfusion.EJ2;  
@using Syncfusion.EJ2.Maps;  
@{  
    var border = new Syncfusion.EJ2.Maps.MapsBorder  
    {  
        Color = "green",
```

```

        Width = 2,
        Opacity = 1
    };
    var margin = new Syncfusion.EJ2.Maps.MapsMargin
    {
        Bottom = 10,
        Top = 10,
        Left = 20,
        Right = 20
    };
}
@Html.EJS().Maps("maps").Background("#CCD1D1").Border(border).Margin(margin)
.Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

MAP-CONTAINER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
            System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```




Customizing Maps area

By default, the background color of the shape maps is set as **white**. To modify the background color of the Maps area, the **Background** property in the **MapsMapsAreaSettings** is used. The border of the Maps area can be customized using the **Border** property in the **MapsMapsAreaSettings** class.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var maparea = new Syncfusion.EJ2.Maps.MapsMapsAreaSettings
    {
        Background = "#CCD1D1",
        Border = new Syncfusion.EJ2.Maps.MapsAreaSettingsBorderMapsArea
        {
            Width = 2,
```

```

        Color = "green",
        Opacity = 1
    }
};
}
@Html.EJS().Maps("maps").MapsArea(maparea).Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

MAP-AREA.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Customizing the shapes

The following properties are available in `MapsShapeSettings` class to customize the shapes of the Maps.

- `Fill` - To apply the fill color to the all the shapes.
- `Autofill` - To apply the palette colors to the shapes if it is set as true.
- `Palette` - To set the custom palette for the shapes.
- `DashArray` - To define the pattern of dashes and gaps that is applied to the outline of the shapes.
- `Opacity` - To customize the transparency for the shapes.
- `Border` - To customize the color, width and opacity of the border of the shapes.

CSHTML

```

@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var shapsettings = new Syncfusion.EJ2.Maps.MapsShapeSettings
    {
        Autofill = true,
        Palette = new string[] { "#C7DE6C", "#59A076", "#88D0BC", "#FEA78C",
"#FFC557" },
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Width = 3,
            Color = "#FEE1DD",
            Opacity = 1
        },
        DashArray = "1",
        Opacity = 0.9
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(shapsettings)
    .ShapeData(ViewBag.worldMap).Add();
}).Render()

```

SHAPE-CUSTOMIZATION.CS

```

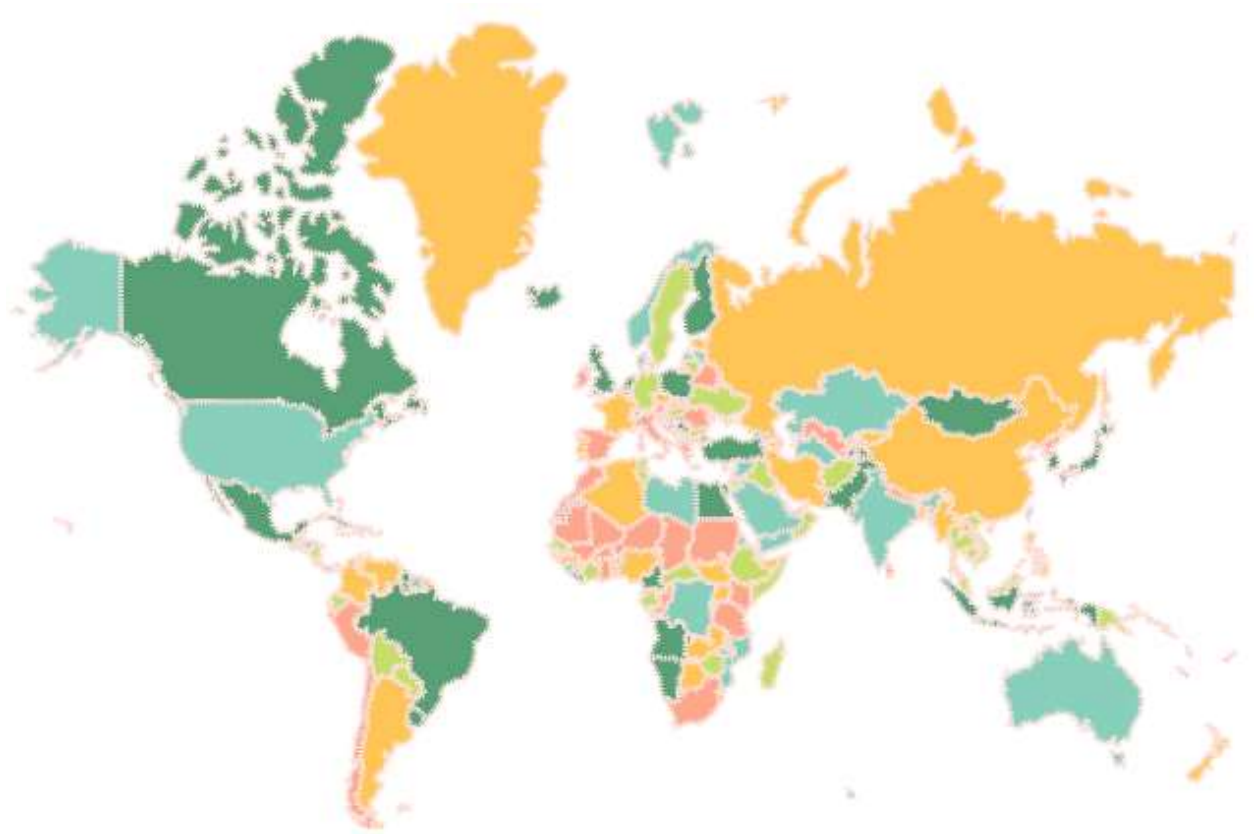
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

```

    }
}
}

```



Setting color to the shapes from the data source

The color for each shape in the Maps can be set using the `ColorValuePath` property of `MapsShapeSettings` class. The value for the `ColorValuePath` property is the field name from the data source of the `MapsShapeSettings` class which contains the color values.

CSHTML

```

@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var data = new[]
    {
        new { continent="North America", color="#71B081"},
        new { continent="South America", color="#5A9A77"},
        new { continent="Africa", color="#498770"},
        new { continent="Europe", color="#39776C"},
        new { continent="Asia", color="#266665"},
        new { continent="Australia", color="#124F5E"}
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss

```

```

        .ColorValuePath("color"))

        .ShapeDataPath("continent").DataSource(data).ShapePropertyPath("continent").
        ShapeData(ViewBag.worldMap).Add();
    }).Render()

```

SHAPE-COLORDS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
            System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Applying border to individual shapes

The border of each shape in the Maps can be customized using the `BorderColorValuePath` and `BorderWidthValuePath` properties to modify the color and the width of the border respectively. The field name in the data source of the layer which contains the color and the width values must be set in the `BorderColorValuePath` and `BorderWidthValuePath` properties respectively. If the values of `BorderWidthValuePath` and `BorderColorValuePath` do not match with the field name from the data source, then the color and width of the border will be applied to the shapes using the `Border` property in the `MapsShapeSettings` class.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var data = new[]
    {
        new { continent="North America", color="#71B081",
borderColor="#CCFFE5", width=2},
        new { continent="South America", color="#5A9A77", borderColor="red",
width=2},
        new { continent="Africa", color="#498770", borderColor="#FFCC99",
width=2},
        new { continent="Europe", color="#39776C", borderColor="#66B2FF",
width=2},
    }
```

```

        new { continent="Asia", color="#266665", borderColor="#999900",
width=2},
        new { continent="Australia", color="#124F5E", borderColor="blue",
width=2}
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss

.ColorValuePath("color").BorderColorValuePath("borderColor").BorderWidthValuePath("width")

.ShapeDataPath("continent").DataSource(data).ShapePropertyPath("continent").ShapeData(ViewBag.worldMap).Add());
}).Render()

```

SHAPE-BORDER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```




Projection type

The Maps component supports the following projection types:

- Mercator
- Equiarectangular
- Miller
- Eckert3
- Eckert5
- Eckert6
- Winkel3
- AitOff

By default, the Maps are rendered by the **Mercator** projection type in which the Maps are rendered based on the coordinates. So, the Maps is not stretched. To change the type of projection in the Maps, the **ProjectionType** property is used.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().Maps("maps").ProjectionType(Syncfusion.EJ2.Maps.ProjectionType.M  
iller).Layers(l =>  
{  
    l.ShapeData(ViewBag.worldMap).Add();  
}).Render()
```

PROJECTION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Color Mapping in ASP.NET MVC Maps Component

Color mapping is used to customize the shape colors based on the given values. It has three types.

1. Range color mapping
2. Equal color mapping
3. Desaturation color mapping

To add color mapping to the shapes of the Maps, bind the data source to the `DataSource` property of `MapsLayer` and set the field name which contains the color value in the data source to the `ColorValuePath` property.

Types of color mapping

Range color mapping

Range color mapping applies the color to the shapes of the Maps which matches the numeric values in the data source within the given color mapping ranges. The **From** and **To** properties in the **MapsColorMapping** are used to mention the color mapping ranges in the Maps.

Bind the **ViewBag.populationData** data to the **DataSource** property of **MapsLayer** and set the **ColorValuePath** property of **MapsShapeSettings** as **density**. The range values can be set using the **From** and **To** properties of **MapsColorMapping**.

```
`sh
[
...
{
'code': 'AE',
'value': 90,
'name': 'United Arab Emirates',
'population': 8264070,
'density': 99
},
{
'code': 'GB',
'value': 257,
'name': 'United Kingdom',
'population': 62041708,
'density': 255
},
{
'code': 'US',
'value': 34,
'name': 'United States',
'population': 325020000,
'density': 33
}
...
];
`
```

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var colorMapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping {From = 0.00001, To = 100, Color =
"rgb(153,174,214)"},
        new MapsColorMapping {From = 100, To = 200, Color =
"rgb(115,143,199)"},
        new MapsColorMapping {From = 200, To = 300, Color =
"rgb(77,112,184)"},
        new MapsColorMapping {From = 300, To = 500, Color =
"rgb(38,82,168)"},
        new MapsColorMapping {From = 500, To = 19000, Color =
"rgb(0,51,153)"}
    };
}
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss.Fill("#E5E5E5").ColorValuePath("density")
        .ColorMapping(colorMapping)).DataSource(ViewBag.populationDensity)

    .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name")
    .Add();
}).Render()

```

COLORVALUEPATH.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Syncfusion.EJ2;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.populationDensity = GetPopulationDensity();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}

```

```
    }  
    public object GetPopulationData()  
    {  
        string text =  
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/Population-  
density.json");  
        return JsonConvert.DeserializeObject(text);  
    }  
    public object GetMap()  
    {  
        string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
        return JsonConvert.DeserializeObject(allText, typeof(object));  
    }  
    public object GetPopulationDensity()  
    {  
        string text =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Population-  
density.json"));  
        return JsonConvert.DeserializeObject(text, typeof(object));  
    }  
}
```



Note: Refer the data values of [Population-density.json](#) here.

Equal color mapping

Equal color mapping applies the color to the shapes of the Maps when the **Value** property of **MapsColorMapping** matches with the values provided in the data source. The following example shows how to apply equal color mapping to the shapes with the data source **unCountries** which illustrates the permanent and non-permanent countries in the UN security council.

```
`sh
[
{ Country: 'China', Membership: 'Permanent' },
{ Country: 'France', Membership: 'Permanent' },
{ Country: 'Russia', Membership: 'Permanent' },
{ Country: 'United Kingdom', Membership: 'Permanent' },
{ Country: 'United States', Membership: 'Permanent' },
{ Country: 'Bolivia', Membership: 'Non-Permanent' },
{ Country: 'Eq. Guinea', Membership: 'Non-Permanent' },
{ Country: 'Ethiopia', Membership: 'Non-Permanent' },
{ Country: "Côte d'Ivoire", Membership: 'Permanent' },
{ Country: 'Kazakhstan', Membership: 'Non-Permanent' },
{ Country: 'Kuwait', Membership: 'Non-Permanent' },
{ Country: 'Netherlands', Membership: 'Non-Permanent' },
{ Country: 'Peru', Membership: 'Non-Permanent' },
{ Country: 'Poland', Membership: 'Non-Permanent' },
{ Country: 'Sweden', Membership: 'Non-Permanent' },
]
```

Bind the **ViewBag.unitedCountries** data to the **DataSource** property of **MapsLayer** and set the **ColorValuePath** property of **MapsShapeSettings** as **Membership**. Set the **Value** property in the **MapsColorMapping** to **Permanent** and **Non-Permanent** in the different set of shape color mapping properties. If the corresponding value of the **ColorValuePath** property matches with the corresponding field name in the data source, then the given color will be applied.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#EDB46F",Value= "Permanent" },
        new MapsColorMapping { Color= "#F1931B", Value = "Non-Permanent" }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
```

```
{
    l.ShapeSettings(ss => ss.Fill("#E5E5E5").ColorValuePath("Membership")
        .ColorMapping(ViewBag.colorData)).DataSource(ViewBag.unitedCountries)

    .ShapeData(ViewBag.worldMap).ShapeDataPath("Country").ShapePropertyPath("name").Add();
})).Render();
```

EQUALCOLORMAPPING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.unCountries = GetUnCountries();
            ViewBag.unitedCountries = GetUnitedCountries();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetUnCountries()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/uncountries.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetWorldMap()
        {
            string text =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetUnitedCountries()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/uncountries.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



```
}
}
```



Desaturation color mapping

Desaturation color mapping applies the color to the shapes of the Maps, similar to the range color mapping. The opacity will be applied in this color mapping based on the **MinOpacity** and **MaxOpacity** properties in the **MapsColorMapping**.

Note: The following example shows how to apply desaturation color mapping to the shapes with the data source **Population_Density** that is available in the [Range color mapping](#) section.

Bind the **Population_Density** data to the **DataSource** property of **MapsLayer** class and set the **ColorValuePath** property of **MapsShapeSettings** as **density**. The range values can be set using the **From** and **To** properties in the **MapsColorMapping**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ From = 1, To = 100, MinOpacity = 0.3,
        MaxOpacity = 1, Color = "rgb(153,174,214)" },
        new MapsColorMapping { From = 101, To = 200, MinOpacity = 0.4,
        MaxOpacity = 1, Color = "rgb(115,143,199)" }
```

```

    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss.Fill("#E5E5E5").ColorValuePath("density")
    .ColorMapping(colormapping)).DataSource(ViewBag.populationDensity)

    .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name")
    .Add();
}).Render()

```

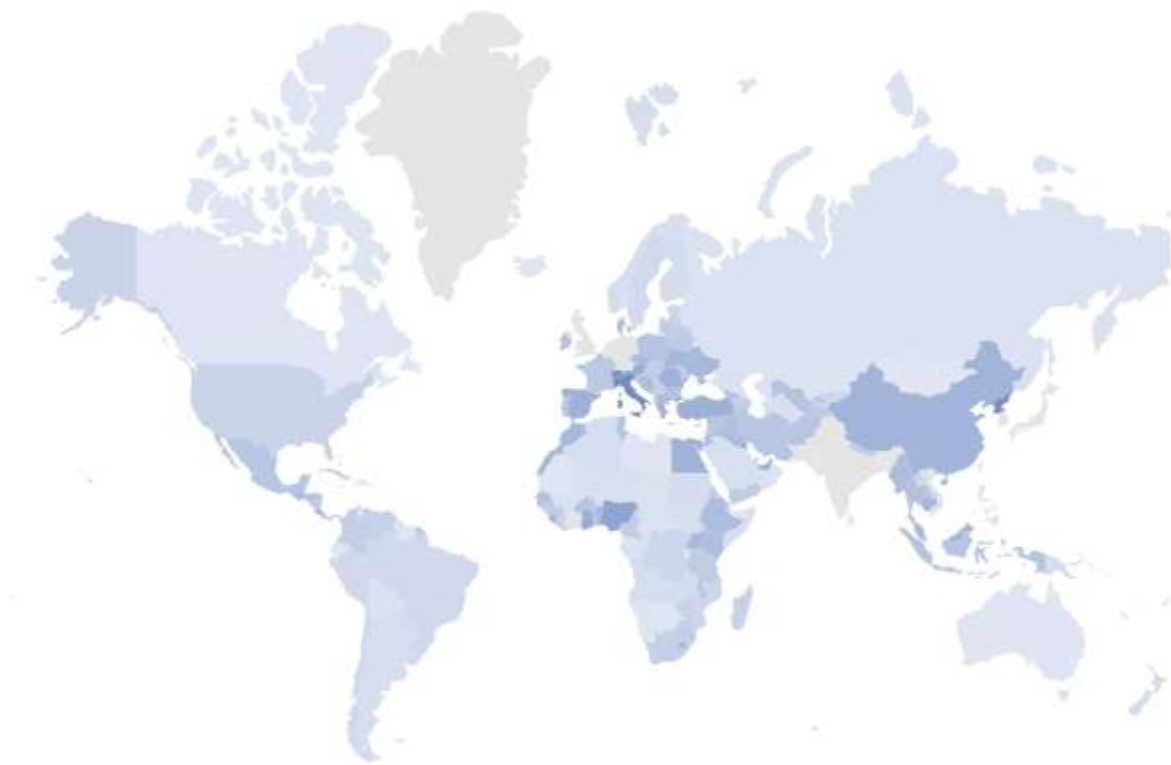
DESATURATIONCOLORMAPPING.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.populationDensity = GetPopulationDensity();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetPopulationData()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/Population-
density.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetPopulationDensity()
        {

```

```
string text =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Population-  
density.json"));  
return JsonConvert.DeserializeObject(text, typeof(object));  
}  
}
```



Refer the data values of [Population-density.json](#) here.

Multiple colors for a single shape

Multiple colors can be added to the color mapping which can be used as gradient effect to a specific shape based on the ranges in the data source. By using the **Color** property of **MapsColorMapping**, any number of colors can be set to the shapes as a gradient.

Note: The following example demonstrates how to use multiple colors in color mapping with the data source **Population_Density** that is available in the [Range color mapping](#) section.

Bind the **Population_Density** data to the **DataSource** property of **MapsLayer** class and set the **ColorValuePath** property of **MapsShapeSettings** as **density**. The range values can be set using the **From** and **To** properties of **MapsColorMapping**.

CSHTML

```
@using Syncfusion.EJ2;
```

```

@using Syncfusion.EJ2.Maps;
@{
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ From = 0, To = 100, Color = new string[]
{"red", "blue"} },
        new MapsColorMapping { From = 101, To = 200, Color = new string[]
{"green", "yellow"} }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss.Fill("#E5E5E5").ColorValuePath("density")
        .ColorMapping(colormapping)).DataSource(ViewBag.populationDensity)

    .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name")
    .Add();
}).Render()

```

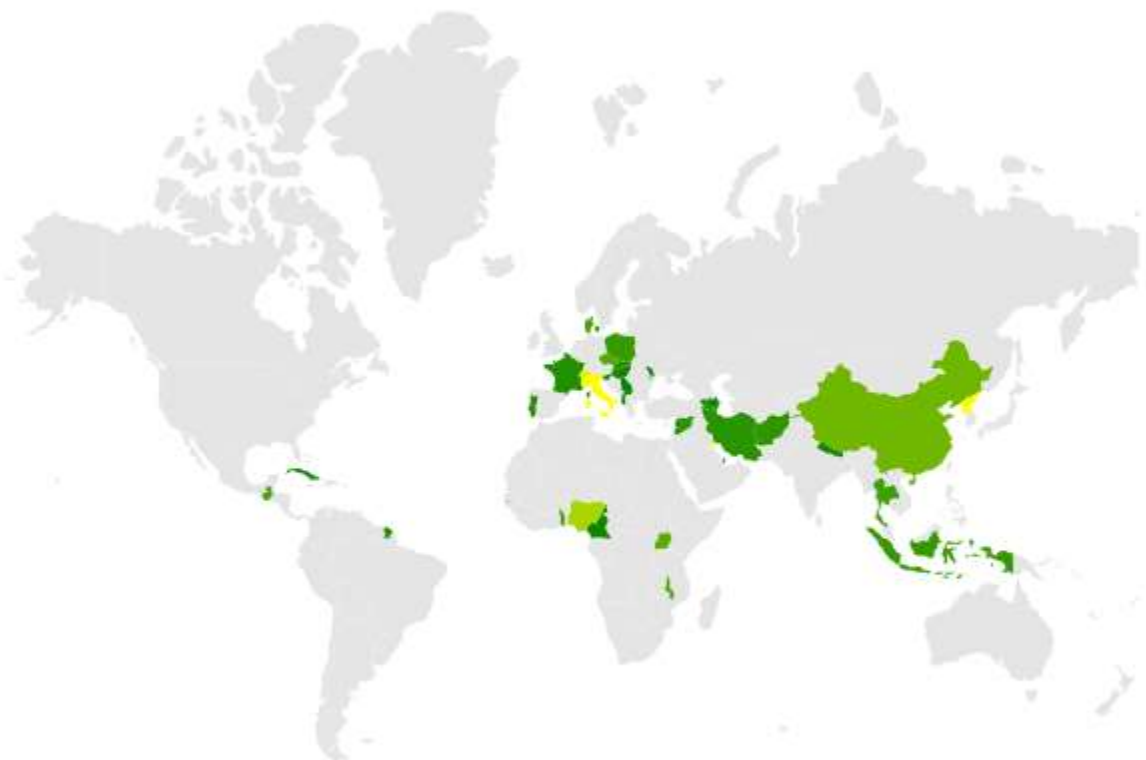
DESATURATIONWITHMULTIPLECOLORS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.populationDensity = GetPopulationDensity();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetPopulationData()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Population-
density.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()

```

```
{
    string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
    return JsonConvert.DeserializeObject(allText, typeof(object));
}
public object GetPopulationDensity()
{
    string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Population-
density.json"));
    return JsonConvert.DeserializeObject(text, typeof(object));
}
}
```



Note: Refer the data values of [Population-density.json](#) here.

Color for items excluded from color mapping

Color mapping can be applied to the shapes in the Maps which does not match color mapping criteria such as range or equal values using the **Color** property of **MapsColorMapping**.

Note: The following example shows how to set the color for items excluded from the color mapping with the data source **Population_Density** that is available in the [Range color mapping](#) section.

In the following example, color mapping is added for the ranges from 0 to 200. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the `Color` property alone in the `MapsColorMapping`.

CSHTML

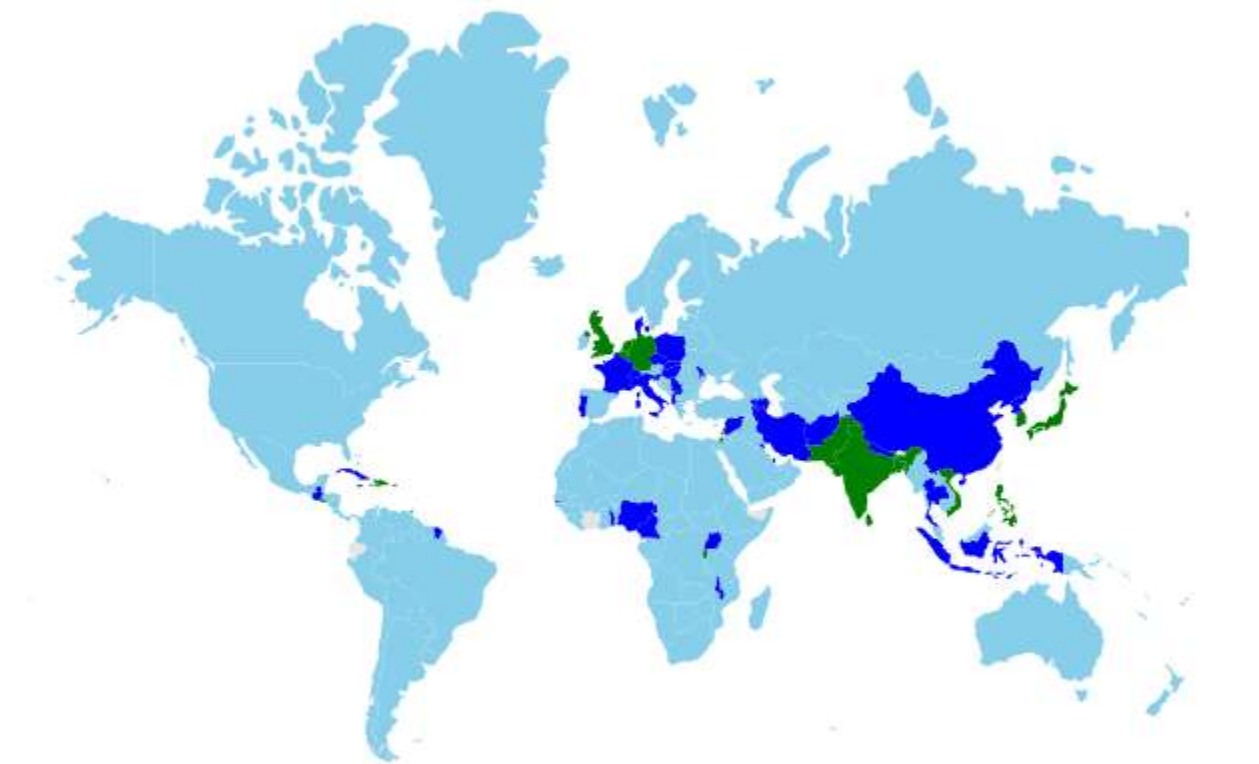
```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ From = 0.001, To = 100, Color = "skyblue" },
        new MapsColorMapping { From = 101, To = 200, Color = "blue" },
        new MapsColorMapping {Color = "green"}
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(ss => ss.Fill("#E5E5E5").ColorValuePath("density")
        .ColorMapping(colormapping)).DataSource(ViewBag.populationDensity)

    .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name")
    .Add();
}).Render()
```

EXCLUDEDCOLORMAPPING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.populationDensity = GetPopulationDensity();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetPopulationData()
        {
            string allText =
                System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Population-
                density.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetWorldMap()
        {
```

```
{
    string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
    return JsonConvert.DeserializeObject(text);
}
public object GetMap()
{
    string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
    return JsonConvert.DeserializeObject(allText, typeof(object));
}
public object GetPopulationDensity()
{
    string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/Population-
density.json"));
    return JsonConvert.DeserializeObject(text, typeof(object));
}
}
```



Note: Refer the data values of [Population-density.json](#) here.

Color mapping for bubbles

The color mapping types such as range color mapping, equal color mapping and desaturation color mapping are applicable for bubbles in the Maps. To add color mapping for bubbles of the Maps, bind the data source to the **DataSource** property of **MapsBubble** class and set the field name which contains the color value in the data source to the **ColorValuePath** property. Multiple colors for a single set of bubbles and color for excluded items from **ColorMapping** are also applicable for bubbles.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new { name= "India", population= "38332521" },
        new { name= "Australia", population= "19651127" },
        new { name= "Pakistan", population= "3090416" }
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping>
    {
        new MapsColorMapping{ Color = "#C3E6ED",Value= "38332521" },
        new MapsColorMapping { Color= "#F1931B", Value = "19651127" },
        new MapsColorMapping { Color= "blue", Value = "3090416" }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.BubbleSettings(bubble =>
    {
        bubble.Visible(true).ValuePath("population").MinRadius(5).MaxRadius(20).ColorValuePath("population").ValuePath("population").
            DataSource(data).ColorMapping(colormapping).Add();
    }).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
}).Render()
```

BUBBLE-COLORMAPPING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
        }
    }
}
```



```
        ViewBag.worldMap = GetMap();  
        return View();  
    }  
    public object GetWorldMap()  
    {  
        string allText =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");  
        return JsonConvert.DeserializeObject(allText);  
    }  
    public object GetMap()  
    {  
        string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
        return JsonConvert.DeserializeObject(allText, typeof(object));  
    }  
}
```



Data labels in ASP.NET MVC Maps Component

Data labels provide information to users about the shapes of the Maps component. It can be enabled by setting the `Visible` property of the `MapsDataLabelSettings` to `true`.

Adding data labels

To display data labels in the Maps, the `LabelPath` property of `MapsDataLabelSettings` must be used. The value of the `LabelPath` property can be taken from the field name in the shape data or data source. In the following example, the value of the `LabelPath` property is the field name in the shape data of the Maps layer.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s => s.Autofill(true)).DataLabelSettings(label =>
label.Visible(true).LabelPath("name")).
    ShapeData(ViewBag.usmap).Add();
}).Render()
```

LABEL.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usmap = GetUnitedStatesMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUnitedStatesMap()
        {
            string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}
```



In the following example, the value of `LabelPath` property is set from the field name in the data source of the layer settings.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var data = new[]
    {
        new {name="Afghanistan", value=53, countryCode="AF",
population=29863010, color="red",
            density=119, continent="Asia"},
        new {name="Albania", value=117, countryCode="AL",
population=3195000, color="Blue",
            density=111, continent="Europe"},
        new {name="Algeria", value=15, countryCode="DZ",
population=34895000, color="Green",
            density=15, continent="Africa"}
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s => s.Autofill(true)).DataLabelSettings(label =>
label.Visible(true).LabelPath("continent").SmartLabelMode(Syncfusion.EJ2.Map
s.SmartLabelMode.Trim)).
        ShapeDataPath("name").ShapePropertyPath("name")
})
```

```
.DataSource(data).ShapeData(ViewBag.worldMap).Add();  
}).Render();
```

LABEL-DS.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            ViewBag.world_map = GetWorldMap();  
            ViewBag.worldMap = GetMap();  
            return View();  
        }  
        public object GetWorldMap()  
        {  
            string allText =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");  
            return JsonConvert.DeserializeObject(allText);  
        }  
        public object GetMap()  
        {  
            string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
            return JsonConvert.DeserializeObject(allText, typeof(object));  
        }  
    }  
}
```



Customization

The following properties are available in the `MapsDataLabelSettings` to customize the data label of the Maps component.

- **Border** - To customize the color, width and opacity for the border of the data labels in Maps.
- **Fill** - To apply the color of the data labels in Maps.
- **Opacity** - To customize the transparency of the data labels in Maps.
- **TextStyle** - To customize the text style of the data labels in Maps.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var label = new MapsDataLabelSettings
    {
        Visible = true,
        LabelPath = "name",
        Border = new MapsBorder
        {
            Color = "green",
            Width = 2,
            Opacity = 1
        },
    },
}
```

```

        Fill = "#CCD1D1",
        Opacity = 0.9,
        TextStyle = new MapsFont
        {
            Color = "blue",
            Size = "10px",
            FontStyle = "Sans-serif",
            FontWeight = "normal"
        }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s => s.Autofill(true)).DataLabelSettings(label).
        ShapeData(ViewBag.usmap).Add();
}).Render()

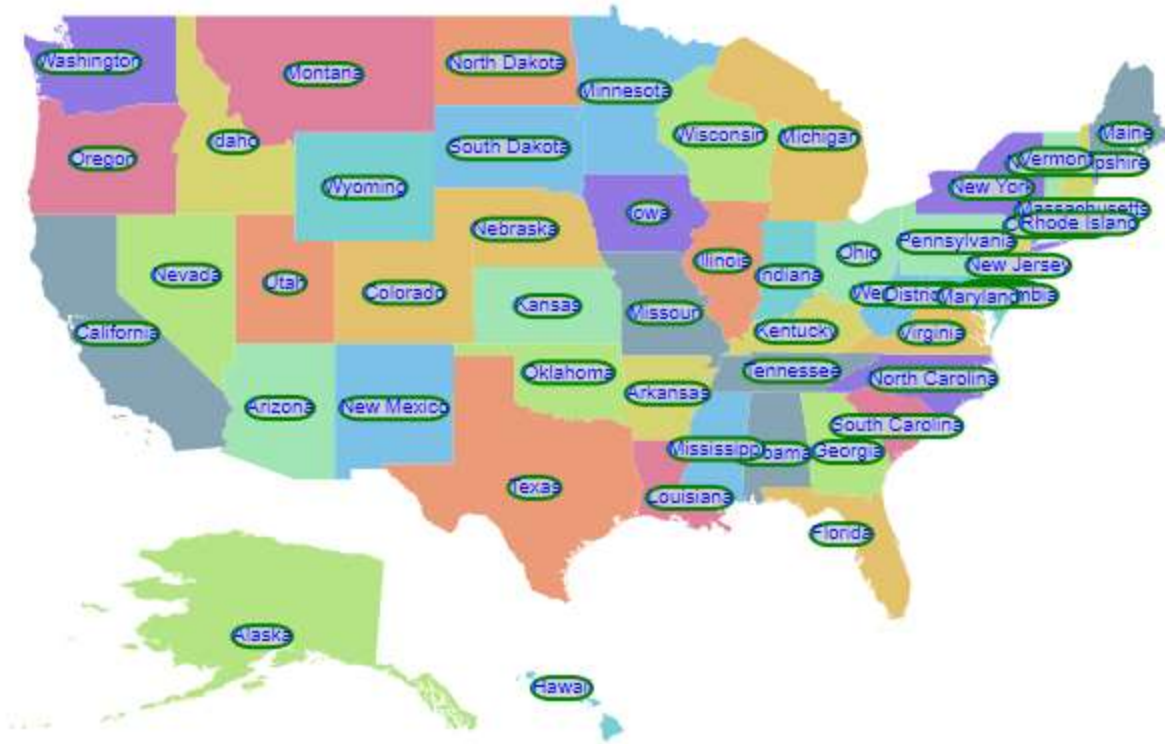
```

LABEL-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usmap = GetUnitedStatesMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUnitedStatesMap()
        {
            string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}

```



Label animation

The data labels can be animated during the initial rendering of the Maps. This can be enabled by setting the [AnimationDuration](#) property in the `DataLabelSettings` of the Maps. The duration of the animation is specified in milliseconds.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var label = new MapsDataLabelSettings
    {
        Visible = true,
        LabelPath = "name",
        IntersectionAction = Syncfusion.EJ2.Maps.IntersectAction.Trim,
        SmartLabelMode = Syncfusion.EJ2.Maps.SmartLabelMode.Hide,
        AnimationDuration = 4000
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s =>
s.Autofill(true)).DataLabelSettings(label).TooltipSettings(ts
=>ts.ValuePath("name").Visible(true)).
        ShapeData(ViewBag.usmap).Add();
}).Render()
```

LABEL-ANIMATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usmap = GetUnitedStatesMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUnitedStatesMap()
        {
            string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}

```

Smart labels

The Maps component provides an option to handle the labels when they intersect with the corresponding shape borders using the **SmartLabelMode** property. The following options are available in the **SmartLabelMode** property.

- None
- Hide
- Trim

CSHTML

```

@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var label = new MapsDataLabelSettings
    {
        Visible = true,
        LabelPath = "name",

```



```

        SmartLabelMode = Syncfusion.EJ2.Maps.SmartLabelMode.Hide
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s => s.Autofill(true)).DataLabelSettings(label).
        ShapeData(ViewBag.usmap).Add();
}).Render()

```

LABEL-MODE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usmap = GetUnitedStatesMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUnitedStatesMap()
        {
            string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}

```



Intersect action

The Maps component provides an option to handle the labels when a label intersects with another label using the `IntersectionAction` property. The following options are available in the `IntersectionAction` property.

- None
- Hide
- Trim

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var label = new MapsDataLabelSettings
    {
        Visible = true,
        LabelPath = "name",
        IntersectionAction = Syncfusion.EJ2.Maps.IntersectAction.Trim
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s => s.Autofill(true)).DataLabelSettings(label).
    ShapeData(ViewBag.usmap).Add();
})
```

```
}).Render()
```

LABEL-INTERSECTION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usmap = GetUnitedStatesMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUnitedStatesMap()
        {
            string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}
```



Adding data label as a template

The data label can be added as a template in the Maps component. The `Template` property of `MapsDataLabelSettings` is used to set the data label as a template. Any text or HTML element can be added as the template in data labels.

Note: The properties of data label such as, `SmartLabelMode`, `IntersectionAction`, `AnimationDuration`, `Border`, `Fill`, `Opacity` and `TextStyle` properties are not applicable to `Template` property. The styles can be applied to the label template using the CSS styles of the HTML element.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var label = new MapsDataLabelSettings
    {
        Visible = true,
        Template = "Label"
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(s => s.Autofill(true)).DataLabelSettings(label).
    ShapeData(ViewBag.usmap).Add();
}).Render()
```

LABEL-TEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usmap = GetUnitedStatesMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/USA.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetUnitedStatesMap()
        {
            string text =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}
```



Polygon shape in ASP.NET MVC Maps component

The Maps component allows you to add polygon shape to a geometry map or an online map by using the properties in the [Polygons](#). This section describes how to add polygon shape to the map and customize them.

The polygon shape can be rendered over the map layer by defining the [Points](#) property in the [Polygons](#) of the Maps component. The [Points](#) property uses a collection of latitude and longitude values to define the polygon shape.

The [Polygons](#) provides the following properties for customizing the polygon shape of the Maps component.

- [Fill](#) - It is used to change the color of the polygon shape.
- [Opacity](#) - It is used to change the opacity of the polygon shape.
- [BorderColor](#) - It is used to change the color of the border in the polygon shape.
- [BorderWidth](#) - It is used to change the width of the border in the polygon shape.
- [BorderOpacity](#) - It is used to change the opacity of the border in the polygon shape.

You can also include “n” polygon shapes using the [Polygons](#) property.

The following example shows how to customize the polygon shape over the geometry map.

CSHTML

```
@using Syncfusion.EJ2;
```

```
@{
    var data = new[]
    {
        new { longitude = 34.88539587371454, latitude = 28.181421087099537 },
        new { longitude = 37.50029619722466, latitude = 24.299419888989462 },
        new { longitude = 39.22241423764024, latitude = 22.638529461838658 },
        new { longitude = 38.95650769309776, latitude = 21.424998160017495 },
        new { longitude = 40.19963938650778, latitude = 20.271205391339606 },
        new { longitude = 41.76547269134551, latitude = 18.315451049867193 },
        new { longitude = 42.78452077838921, latitude = 16.097235052947966 },
        new { longitude = 43.36984949591576, latitude = 17.188572054533054 },
        new { longitude = 44.12558191797012, latitude = 17.407258102232234 },
        new { longitude = 46.69027032797584, latitude = 17.33342243475734 },
        new { longitude = 47.09312386141585, latitude = 16.97087769526452 },
        new { longitude = 48.3417299826302, latitude = 18.152700711188004 },
        new { longitude = 49.74762591400318, latitude = 18.81544363931681 },
        new { longitude = 52.41428026336621, latitude = 18.9035706497573 },
        new { longitude = 55.272683129240335, latitude = 20.133861012918544 },
        new { longitude = 55.60121336079203, latitude = 21.92042703112351 },
        new { longitude = 55.08204399107967, latitude = 22.823302662258882 },
        new { longitude = 52.743894337844154, latitude = 22.954463486477437 },
        new { longitude = 51.47035908651375, latitude = 24.35818837668566 },
        new { longitude = 51.122553219055874, latitude = 24.666679732426346 },
        new { longitude = 51.58731671256831, latitude = 25.173806925822717 },
        new { longitude = 51.35950585992913, latitude = 25.84556484481108 },
        new { longitude = 51.088770529661275, latitude = 26.168494193631147 },
        new { longitude = 50.78527056538036, latitude = 25.349051242147596 },
        new { longitude = 50.88330288802666, latitude = 24.779242606720743 },
        new { longitude = 50.19702490702369, latitude = 25.66825106363693 },
        new { longitude = 50.066461167339924, latitude = 26.268905608606616 },
        new { longitude = 49.645896067213215, latitude = 27.15116474192905 },
        new { longitude = 48.917371072320805, latitude = 27.55738830340198 },
        new { longitude = 48.3984720209192, latitude = 28.566207269716173 },
        new { longitude = 47.68851714518985, latitude = 28.5938991332588 },
        new { longitude = 47.45059089191449, latitude = 29.009321449856984 }
    },
}
```

```

        new { longitude = 44.73549453609391, latitude = 29.157358362696385
    },
    new { longitude = 41.79487372890989, latitude = 31.23489959729713 },
    new { longitude = 40.36977176033773, latitude = 31.9642352513131 },
    new { longitude = 39.168270913149826, latitude = 32.18348471414393
    },
    new { longitude = 37.019253492546454, latitude = 31.47710220862595
    },
    new { longitude = 37.99644645508337, latitude = 30.4851028633376 },
    new { longitude = 37.67756530485232, latitude = 30.3636358598429 },
    new { longitude = 37.50181466030105, latitude = 29.960155516804974
    },
    new { longitude = 36.700288181129594, latitude = 29.882136586478993
    },
    new { longitude = 36.100009274845206, latitude = 29.15308642012721
    },
    new { longitude = 34.85774476486728, latitude = 29.3103032832622 },
    new { longitude = 34.64498583263142, latitude = 28.135787235699823
    },
    new { longitude = 34.88539587371454, latitude = 28.181421087099537 }
    };
    var polygons = new List<Syncfusion.EJ2.Maps.MapsPolygon>
    {
        new Syncfusion.EJ2.Maps.MapsPolygon{ Points=data, Fill="blue",
        Opacity=0.7, BorderColor="green", BorderOpacity=0.7, BorderWidth=2 }
    };
    @ (Html.EJS().Maps("maps").Layers(layers => { layers.PolygonSettings(polygon
=> { polygon.Polygons(polygons); }).ShapeData(ViewBag.world_map).Add();
}).Render())

```

POLYGON-SHAPE.CS

```

using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        // To access the data in Core
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        // To access the data in MVC
        public object GetMap()
        {

```



```
string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
return JsonConvert.DeserializeObject(allText, typeof(object));  
}  
}
```



Markers

Markers are notes that are used to leave a message on the Maps. It indicates or marks a specific location with desired symbols on the Maps. It can be enabled by setting the **Visible** property of the **MapsMarker** to **true**.

Adding marker

To add the markers, the **DataSource** property of the **MapsMarker** has a list of objects that contains the data for markers. Using this property, any number of markers can be added to the layers of the Maps. By default, it displays the markers based on the specified latitude and longitude in the given data source. Each data source object contains the following list of properties.

- latitude - The latitude point which determines the X location of the marker.
- longitude - The longitude point which determines the Y location of the marker.

CSHTML

```
@using Syncfusion.EJ2;  
@using Syncfusion.EJ2.Maps;
```

```
@{
    var propertyPath = new[] { "name" };
    var data = new[]
    {
        new {latitude = 49.95121990866204, longitude = 18.468749999999998},
        new {latitude = 59.88893689676585, longitude = -109.3359375},
        new {latitude = -6.64607562172573, longitude = -55.546874999999999}
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.MarkerSettings(marker =>
    {
        marker.Visible(true).AnimationDuration(0).DataSource(data).Height(20).Width(
        20).Add();
        }).ShapeData(ViewBag.world_map).Add();
    }).Render()
```

MARKER-ADDING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Adding marker template

The Marker can be added as a template in the Maps component. The `Template` property of the `MapsMarker` is used to set the HTML element or id of an element as a template.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l=> {
    l.ShapeData( ViewBag.world_map).MarkerSettings(new List<MapsMarker>
    {
        new MapsMarker{Visible=true, Template="#template", DataSource= new[]
        {
            new {latitude= 37.0000, longitude= -120.0000, city= "California"
        },
            new {latitude= 40.7127, longitude= -74.0059, city= "New York" },
            new {latitude= 42.0000, longitude= -93.0000, city= "Iowa" }
        }
    }).Add();
}).Render()
<div id="template" style="display: none;">
    <div>
```

```

<div style="margin-left:8px;height:45px;width:120px;margin-top:-
23px;">
    <label style="color:black;margin-left:15px;font-
weight:normal;">\{\{\:city\}\}</label>
    </div>
</div>
</div>

```

MARKER-LABEL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Customization

The following properties are available in `MapsMarker` class to customize the Markers of the Maps component.

- **Border** - To customize the color, width and opacity of the border for the markers in Maps.
- **Fill** - To apply the color for markers in Maps.
- **DashArray** - To define the pattern of dashes and gaps that is applied to the outline of the markers in Maps.
- **Height** - To customize the height of the markers in Maps.
- **Width** - To customize the width of the markers in Maps.
- **Offset** - To customize the position of the markers in Maps.
- **Opacity** - To customize the transparency of the markers in Maps.
- **AnimationDelay** - To change the time delay in the transition for markers.
- **AnimationDuration** - To change the time duration of animation for markers.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {
        new {latitude = 37.0000, longitude = -120.0000, city= "California"},
        new {latitude = 40.7127, longitude = -74.0059, city= "New York"},
    }
}
```

```

        new {latitude = 42.000, longitude = -93.000, city= "Iowa"}
    };
    var border = new MapsBorder
    {
        Color = "green",
        Width = 2,
        Opacity = 1
    };
}
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
}).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeSettings = new MapsShapeSettings
        {
            Fill = "#C1DFF5"
        },
        ShapeData = ViewBag.world_map,
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                Fill = "red",
                DataSource = data,
                DashArray = "1",
                Shape=MarkerType.Balloon,
                TooltipSettings = new MapsTooltipSettings {
                    Visible = true,
                    ValuePath= "area",
                },
                Height= 20,
                Width= 20,
                AnimationDuration= 0,
                AnimationDelay = 100,
                Border = border
            }
        }
    }
}).Render()

```

MARKER-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers

```

```
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            ViewBag.worldmap = GetWorldMap();  
            ViewBag.world_map = GetMap();  
            return View();  
        }  
        public object GetWorldMap()  
        {  
            string allText =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");  
            return JsonConvert.DeserializeObject(allText);  
        }  
        public object GetMap()  
        {  
            string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
            return JsonConvert.DeserializeObject(allText, typeof(object));  
        }  
    }  
}
```



Marker shapes

The Maps component supports the following marker shapes. To set the shape of the marker, the **Shape** property in **MapsMarker** is used.

- Balloon
- Circle
- Cross
- Diamond
- Image
- Rectangle
- Start
- Triangle
- VerticalLine
- HorizontalLine

Rendering marker shape as image

To render a marker as an image in Maps, set the **Shape** property of **MapsMarker** as **Image** and specify the path of the image to **ImageUrl** property. There is another way to render a marker as an image using the **ImageUrlValuePath** property of the **MapsMarker**. Bind the field name that contains the path of the image in the data source to the **ImageUrlValuePath** property.

CSHTML

```
using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{ var data = new[]
    {
        new {latitude = 37.0000, longitude = -120.0000, city= "California"},
        new {latitude = 40.7127, longitude = -74.0059, city= "New York"},
        new {latitude = 42.000, longitude = -93.000, city= "Iowa"}
    }; }
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.worldMap,
        ShapeSettings = new MapsShapeSettings
        {
            Fill="lightgray"
        },
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
    {
        new Syncfusion.EJ2.Maps.MapsMarker
        {
            Visible = true,
            Fill = "green",
            DataSource = data,
            Shape=MarkerType.Image,
            Height= 20,
            Width= 20,
            ImageUrl="~/ballon.png"
        }
    }
}).Render()
```

MARKER-IMAGE.CS


```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Multiple marker groups

Multiple groups of markers can be added in the Maps by adding multiple **MapsMarker** in the **MapsMarkers** and customization for the markers can be done with the **MapsMarker**.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {
        new {latitude = 37.0000, longitude = -120.0000, city= "California"},
        new {latitude = 40.7127, longitude = -74.0059, city= "New York"},
        new {latitude = 42.000, longitude = -93.000, city= "Iowa"}
    };
    var data1 = new[]
    {
        new {latitude = 19.228825, longitude = 72.854118, city= "Mumbai"},
        new {latitude = 28.610001, longitude = 77.230003, city= "Delhi"},
        new {latitude = 13.067439, longitude = 80.237617, city= "Chennai"}
    };
}
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
}).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
```

```

        new Syncfusion.EJ2.Maps.MapsLayer
        {
            ShapeData = ViewBag.world_map,
            MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
            {
                new Syncfusion.EJ2.Maps.MapsMarker
                {
                    Visible = true,
                    Fill = "green",
                    DataSource = data,
                    Shape=MarkerType.Diamond,
                    Height= 15,
                    Width= 15
                },
                new Syncfusion.EJ2.Maps.MapsMarker
                {
                    Visible = true,
                    Fill = "green",
                    DataSource = data1,
                    Shape=MarkerType.Circle,
                    Height= 15,
                    Width= 15
                }
            }
        }
    }
}).Render()

```

MARKER-GROUP.CS

```

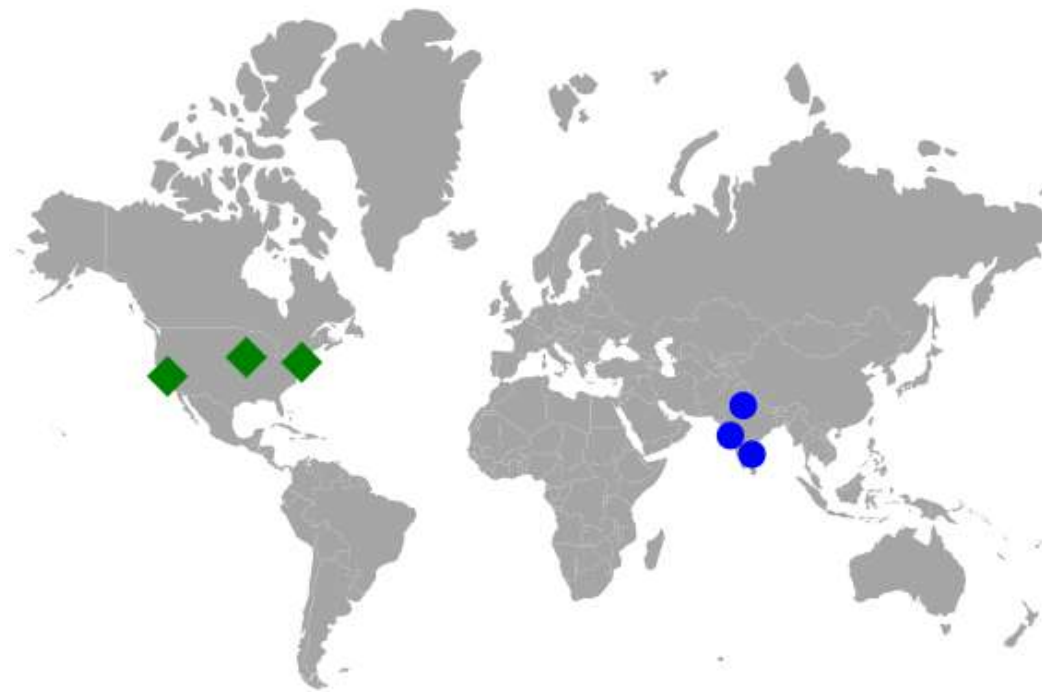
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {

```

```

        string allText =
        System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



Customize marker shapes from data source

Bind different colors and shapes to the marker from data source

Using the **ShapeValuePath** and **ColorValuePath** properties, the color and shape of the marker can be applied from the given data source. Bind the data source to the **DataSource** property of the **MapsMarker** class and set the field names that contains the shape and color values in the data source to the **ShapeValuePath** and **ColorValuePath** properties.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {
        new {latitude = 49.95121990866204, longitude = 18.468749999999998,
        name="Europe", color="red",
        shape="Triangle"},
        new {latitude = 59.88893689676585, longitude = -109.3359375,
        name="North America", color="blue",

```

```

        shape="Pentagon"},
        new {latitude = -6.64607562172573, longitude = -55.54687499999999,
name="South America", color="green",
        shape="InvertedTriangle"}
    };
}
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.world_map,
        MarkerSettings = new List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data,
                ShapeValuePath = "shape",
                ColorValuePath = "color"
            }
        }
    }
}).Render()

```

SHAPE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using EJ2MVCSampleBrowser.Models;
namespace EJ2MVCSampleBrowser.Controllers.Maps
{
    public partial class MapsController : Controller
    {
        // GET: MarkerClustering
        public ActionResult MarkerClustering()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));

```

```

        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



Setting value path from the data source

The latitude and longitude values are used to determine the location of each marker in the Maps. The `LatitudeValuePath` and `LongitudeValuePath` properties are used to specify the value path that presents in the data source of the marker. In the following example, the field name from the data source is set to the `LatitudeValuePath` and `LongitudeValuePath` properties.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {
        new {latitude = 49.95121990866204, longitude = 18.468749999999998},
        new {latitude = 59.88893689676585, longitude = -109.3359375},
        new {latitude = -6.64607562172573, longitude = -55.546874999999999}
    };
}
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {

```

```

        ShapeData = ViewBag.world_map,
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
    {
        new Syncfusion.EJ2.Maps.MapsMarker
        {
            Visible = true,
            LatitudeValuePath = "latitude",
            LongitudeValuePath = "longitude",
            DataSource = data
        }
    }
}).Render()

```

MARKER-VALUEPATH.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Repositioning the marker using drag and drop

The markers on the map can be dragged and dropped to change their position. To enable marker drag and drop, set the [EnableDrag](#) property to **true** in the [MarkerSettings](#) property.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {
        new { latitude = 49.95121990866204, longitude = 18.468749999999998,
name= "MarkerOne" },
        new { latitude = 59.88893689676585, longitude = -109.3359375,
name="MarkerTwo"},
        new { latitude = -6.64607562172573, longitude = -55.546874999999999 ,
name="MarkerThree"},
        new { latitude = 23.644385824912135, longitude= 77.83189239539234 ,
name= "MarkerFour"},
        new { latitude = 63.66569332894224, longitude= 98.2225173953924 ,
name= "MarkerFive"}
    };
    var border = new MapsBorder
```



```

        {
            Color = "#285255",
            Width = 2,
            Opacity = 1
        };
    }
}
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.world_map,
        ShapeSettings = new MapsShapeSettings {
            Fill="#C3E6ED"
        },
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
    {
        new Syncfusion.EJ2.Maps.MapsMarker
        {
            Visible = true,
            EnableDrag = true,
            AnimationDuration= 0,
            Shape= MarkerType.Balloon,
            Border = border,
            TooltipSettings = new MapsTooltipSettings {
                Visible = true,
                ValuePath = "name"
            },
            Width= 20,
            Height= 20,
            DataSource = data
        }
    }
}).Render()

```

MARKER-DRAGANDDROP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()

```

```

    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}
}

```



● MarkerOne ● MarkerTwo ● MarkerThree ● MarkerFour ● MarkerFive

Marker zooming

The Maps can be initially scaled to the center value based on the marker distance. This can be achieved by setting the `ShouldZoomInitially` property in `MapsZoomSettings` as **true**.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {

```

```

        new {latitude= 49.95121990866204, longitude= 18.468749999999998,
name= "Europe" },
        new {latitude= 59.88893689676585, longitude= -109.3359375, name=
"North America" },
        new {latitude= -6.64607562172573, longitude= -55.54687499999999,
name= "South America" }
    };
}
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true,
    ShouldZoomInitially = true,
    HorizontalAlignment = Alignment.Near
}).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.world_map,
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data
            }
        }
    }
}).Render()

```

MARKER-ZOOM.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()

```

```
{
    string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
    return JsonConvert.DeserializeObject(allText, typeof(object));
}
}
```



Marker clustering

Maps provide support to cluster the markers when they overlap each other. The number on a cluster indicates how many overlapped markers it contains. If zooming is performed on any of the cluster locations in Maps, the number on the cluster will decrease, and the individual markers will be seen on the map. When zooming out, the overlapping marker will increase. So that it can cluster again and increase the count over the cluster.

To enable clustering in markers, set the `AllowClustering` property of `MapsMarkerClusterSettings` as **true** and customization of clustering can be done with the `MapsMarkerClusterSettings`.

CSHTML

```
@using Syncfusion.EJ2
```

```

@using Syncfusion.EJ2.Maps
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
}).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeSettings = new MapsShapeSettings
        {
            Fill = "#C1DFF5"
        },
        ShapeData = ViewBag.shape_Data,
        MarkerClusterSettings = new
MapsMarkerClusterSettings
        {
            AllowClustering = true,
            Shape = MarkerType.Circle,
            Height = 40,
            Width = 40,
        },
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = ViewBag.Cluster_data,
                Shape=MarkerType.Circle,
                TooltipSettings = new MapsTooltipSettings {
                    Visible = true,
                    ValuePath= "area",
                },
                Height= 20,
                Width= 20,
                AnimationDuration= 0,
            }
        }
    }
}).Render()

```

MARKER-CLUSTER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using EJ2MVCSampleBrowser.Models;
namespace EJ2MVCSampleBrowser.Controllers.Maps
{
    public partial class MapsController : Controller

```

```
{
    // GET: MarkerClustering
    public ActionResult MarkerClustering()
    {
        ViewBag.shapeData = this.GetWorldMap();
        ViewBag.shape_Data = this.GetWMap();
        ViewBag.ClusterData = this.ClusterData();
        ViewBag.Cluster_data = this.getData();
        return View();
    }
    public object ClusterData()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/MapData/ClusterData.js
on"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetWorldMap()
    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object getData()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("../wwwroot/scripts/MapsData/Cluste
rData.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetWMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}
```



Customization of marker cluster

The following properties are available to customize the marker clustering in the Maps component.

- **Border** - To customize the color, width and opacity of the border of cluster in Maps.
- **ConnectorLineSettings** - To customize the connector line in cluster separating the markers.
- **DashArray** - To customize the dash array for the marker cluster in Maps.
- **Fill** - Applies the color of the cluster in Maps.
- **Height** - To customize the height of the marker cluster in Maps.
- **ImageUrl** - To customize the URL path for the marker cluster when the cluster shape is set as image in Maps.
- **LabelStyle** - To customize the text in marker cluster.
- **Offset** - To customize the offset position for the marker cluster in Maps.
- **Opacity** - To customize the opacity of the marker cluster.
- **Shape** - To customize the shape for the cluster of markers.
- **Width** - To customize the width of the marker cluster in Maps.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
```

```

    }).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
    {
        new Syncfusion.EJ2.Maps.MapsLayer
        {
            ShapeData = ViewBag.worldmap,
            MarkerClusterSettings = new MapsMarkerClusterSettings
            {
                AllowClustering = true,
                Height = 40,
                Width = 40,
                Fill = "green",
                Opacity = 0.9,
                AllowClusterExpand = true,
                Shape = Syncfusion.EJ2.Maps.MarkerType.Circle,
                LabelStyle = new MapsFont
                {
                    Color = "White"
                },
                ConnectorLineSettings = new
MapsConnectorLineSettings
                {
                    Color = "orange",
                    Opacity = 0.8,
                    Width = 2
                }
            },
            MarkerSettings = new List<Syncfusion.EJ2.Maps.MapsMarker>
            {
                new Syncfusion.EJ2.Maps.MapsMarker
                {
                    Visible = true,
                    DataSource = ViewBag.ClusterData,
                    Shape= MarkerType.Balloon,
                    Height = 20,
                    Width = 20,
                    AnimationDuration = 0,
                }
            }
        }
    }).Render()

```

CLUSTER-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using EJ2MVCSampleBrowser.Models;
namespace EJ2MVCSampleBrowser.Controllers.Maps
{
    public partial class MapsController : Controller
    {
        // GET: MarkerClustering

```



```

public ActionResult MarkerClustering()
{
    ViewBag.shapeData = this.GetWorldMap();
    ViewBag.ClusterData = this.ClusterData();
    return View();
}
public object ClusterData()
{
    string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/MapData/ClusterData.js
on"));
    return JsonConvert.DeserializeObject(allText, typeof(object));
}
public object GetWorldMap()
{
    string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
    return JsonConvert.DeserializeObject(allText);
}
}

```



Expanding the marker cluster

The cluster is formed by grouping an identical and non-identical marker from the surrounding area. By clicking on the cluster and setting the `AllowClusterExpand` property in `MapsMarkerClusterSettings` as **true** to expand the identical markers. If zooming is performed in any of the locations of the cluster,

the number on the cluster will decrease and the overlapping marker will be split into an individual marker on the map. When performing zoom out, it will increase the marker count and then cluster it again.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var data = new[]
    {
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 49.95121990866204, longitude= 18.468749999999998,
name="Europe" },
        new { latitude= 59.88893689676585, longitude= -109.3359375,
name="North America" },
        new { latitude= -6.64607562172573, longitude= -55.54687499999999,
name="South America"}
    };
}
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
}).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.worldmap,
        MarkerClusterSettings = new MapsMarkerClusterSettings
        {
            AllowClustering = true,
            AllowClusterExpand = true,
            Height = 40,
            Width = 40,
            LabelStyle = new MapsFont
            {
                Color = "White"
            }
        },
        MarkerSettings = new List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data
            }
        }
    }
}
```

```

    }
    } } } .Render()

```

CLUSTER.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using EJ2MVCSampleBrowser.Models;
namespace EJ2MVCSampleBrowser.Controllers.Maps
{
    public partial class MapsController : Controller
    {
        // GET: MarkerClustering
        public ActionResult MarkerClustering()
        {
            ViewBag.shapeData = GetWorldMap();
            ViewBag.clusterdata = GetClusterData();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetClusterData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/ClusterData.js");
            return JsonConvert.DeserializeObject(text, typeof(object));
        }
    }
}

```



Tooltip for marker

Tooltip is used to display more information about a marker on mouse over or touch end event. This can be enabled separately for marker by setting the **Visible** property of **MapsTooltipSettings** to **true**. The **ValuePath** property in the **MapsTooltipSettings** takes the field name that presents in **dataSource** and displays that value as tooltip text.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@{
    var border = new MapsBorder
    {
        Width = 2,
        Color = "green",
        Opacity = 1
    };
    var data = new[]
    {
        new { latitude= 40.7424509, longitude= 1-74.0081468, name="New York"
    }
    };
}
@Html.EJS().Maps("container").ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
```

```

    }).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
    {
        new Syncfusion.EJ2.Maps.MapsLayer
        {
            ShapeData = ViewBag.usamap,
            MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
            {
                new Syncfusion.EJ2.Maps.MapsMarker
                {
                    Visible = true,
                    Fill = "white",
                    DataSource = data,
                    Shape=MarkerType.Circle,
                    TooltipSettings = new MapsTooltipSettings {
                        Visible = true,
                        ValuePath= "name",
                    },
                    Width= 2,
                    AnimationDuration= 0,
                    Border = border
                }
            }
        }
    }).Render()

```

MARKER-TOOLTIP.CS

```

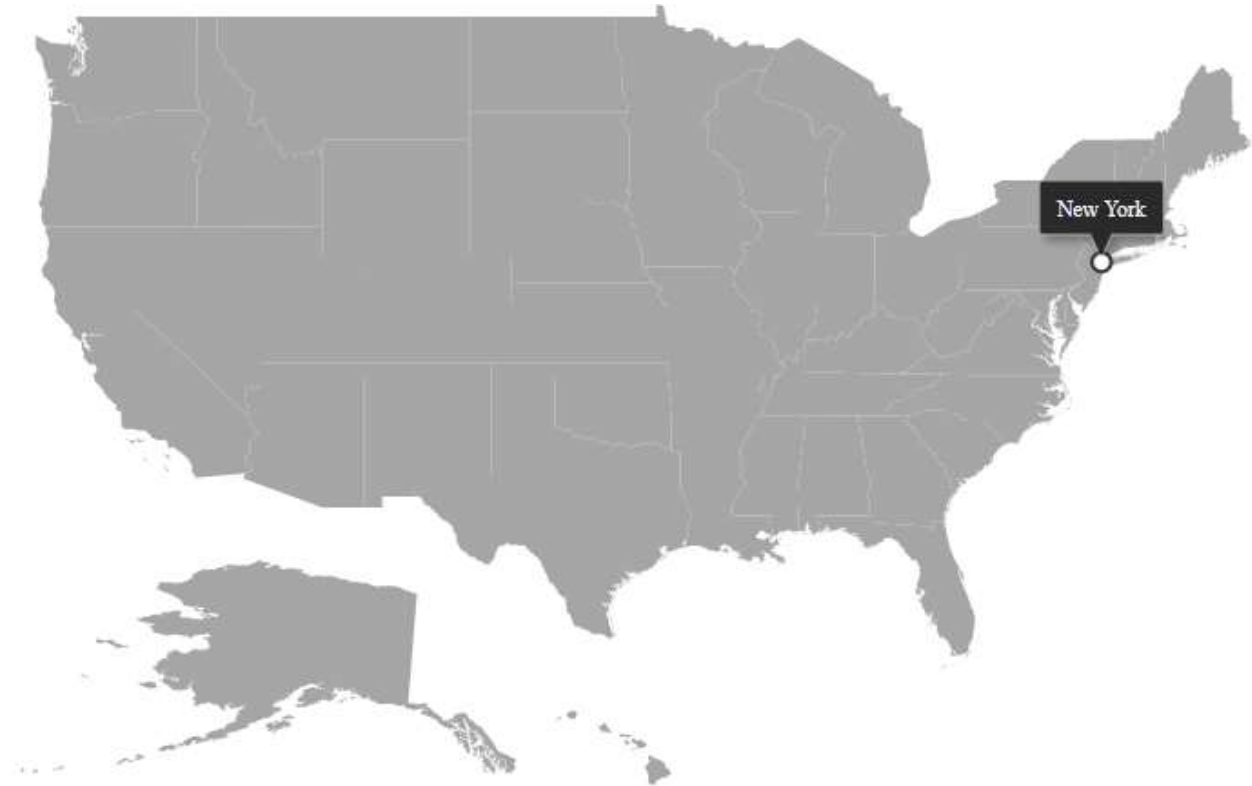
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.usamap = GetUSAMap();
            return View();
        }
        public object GetUSMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetUSAMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));

```

```

        return JsonConvert.DeserializeObject(text);
    }
}

```



Bubbles in ASP.NET MVC Maps Component

Bubbles in the Maps component represent the underlying data values of the Maps. It can be scattered throughout the Maps shapes that contain values in the data source. Bubbles are enabled by setting the **Visible** property of **MapsBubble** to **true**. To add bubbles to the Maps, bind the data source to the **DataSource** property of **MapsBubble** and set the field name, that contains the numerical data, in the data source to the **ValuePath** property.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.BubbleSettings(bubble =>
    {
        bubble.Visible(true).MinRadius(20).MaxRadius(40).ValuePath("population").DataSource(ViewBag.bubbleData).Add();
    })
    .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
})
.Render()

```

BUBBLE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "India", population= "38332521" },
                new BubbleData { name= "Australia", population= "383521" },
                new BubbleData { name= "Pakistan", population= "3090416" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public class BubbleData
        {
            public string name { get; set; }
            public string population { get; set; }
        }
    }
}
```



Bubble shapes

The following types of shapes are available to render the bubbles in Maps.

- Circle
- Square

By default, bubbles are rendered in the **Circle** type. To change the type of the bubble, set the **BubbleType** property of **MapsBubble** as **Square** to render the square shape bubbles.

CSHTML

```
@using Syncfusion.EJ2;  
@using Syncfusion.EJ2.Maps;  
@Html.EJS().Maps("maps").Layers(l =>  
{  
    l.BubbleSettings(bubble =>  
    {  
  
        bubble.Visible(true).ValuePath("population").BubbleType(Syncfusion.EJ2.Maps.  
            BubbleType.Square).DataSource(ViewBag.bubbleData).Add();  
  
    }).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();  
    }).Render()
```


BUBBLE-SHAPE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "India", population= "38332521" },
                new BubbleData { name= "Pakistan", population= "3090416" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public class BubbleData
        {
            public string name { get; set; }
            public string population { get; set; }
        }
    }
}
```



Customization

The following properties are available in **MapsBubble** to customize the bubbles of the Maps component.

- **Border** – To customize the color, width and opacity of the border of the bubbles in Maps.
- **Fill** – To apply the color for bubbles in Maps.
- **Opacity** – To apply opacity to the bubbles in Maps.
- **AnimationDelay** - To change the time delay in the transition for bubbles.
- **AnimationDuration** - To change the time duration of animation for bubbles.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new { name= "Australia", population= "383521" },
        new { name= "Pakistan", population= "3090416" },
        new { name= "Russia", population= "30916" }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.BubbleSettings(bubble =>
```

```

    {

bubble.Visible(true).ValuePath("population").MinRadius(10).MaxRadius(30).ValuePath("population").

DataSource(data).Fill("green").Opacity(1).AnimationDelay(100).AnimationDuration(1000).Border(border => border.Color("blue").Width(2).Opacity(1)).Add();

}).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
}).Render()

```

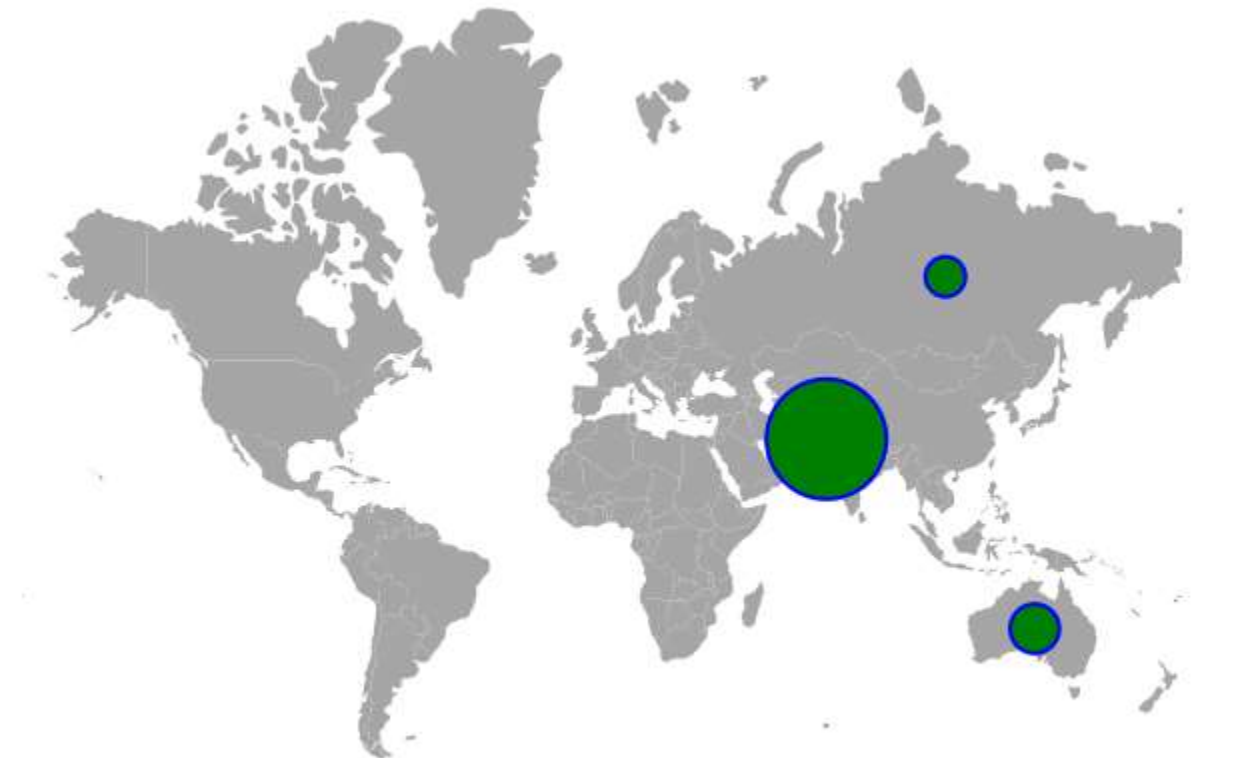
BUBBLE-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "Australia", population= "383521" },
                new BubbleData { name= "Pakistan", population= "3090416" },
                new BubbleData { name= "Russia", population= "30916" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public class BubbleData
        {
            public string name { get; set; }
            public string population { get; set; }
        }
    }
}

```

```
}
}
```



Setting colors to the bubbles from the data source

The color for each bubble in the Maps can be set using the `ColorValuePath` property of `MapsBubble`. The value for the `ColorValuePath` property is the field name from the data source of the `MapsBubble` which contains the color values.

CSHTML

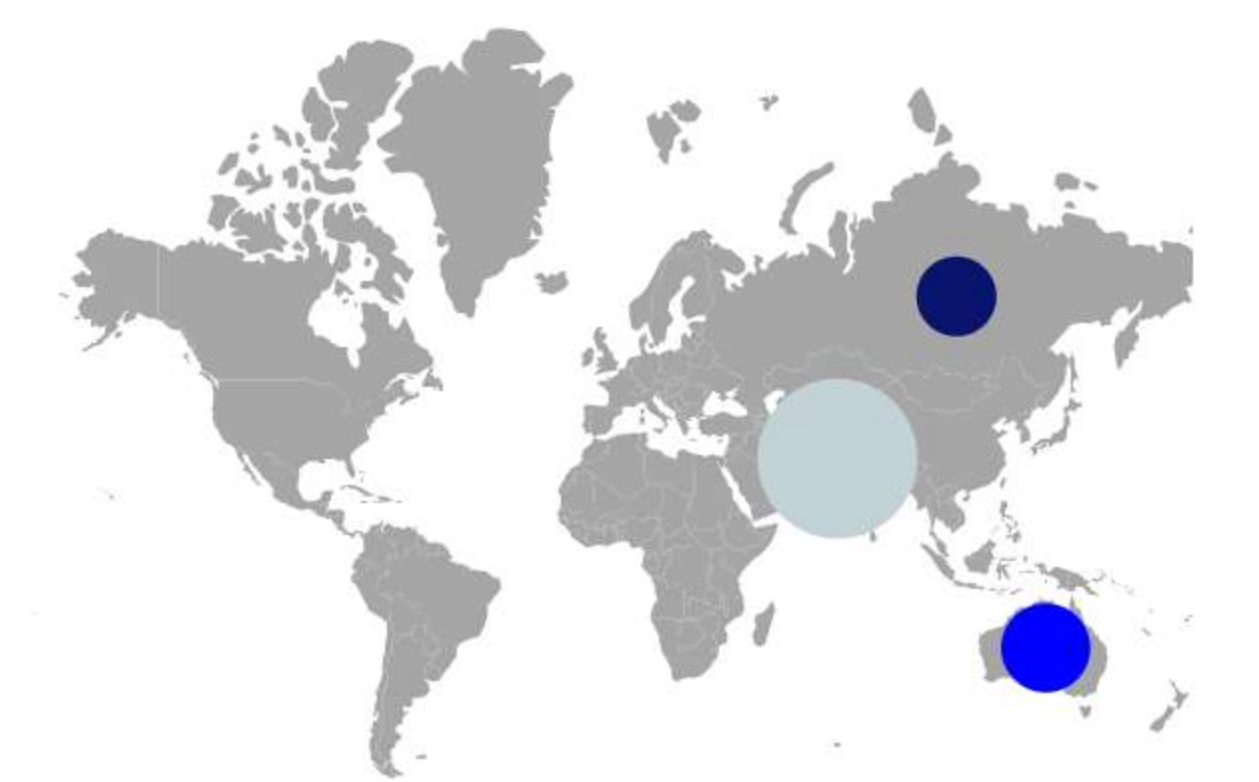
```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.BubbleSettings(bubble =>
    {
        bubble.Visible(true).ValuePath("population").MinRadius(20).MaxRadius(40).ColorValuePath("color").ValuePath("population").
            DataSource(ViewBag.bubbleData).Add();
    }).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
    }).Render()
```

BUBBLE-VALUEPATH.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "Australia", population= "383521",
color="blue" },
                new BubbleData { name= "Pakistan", population= "3090416",
color="#c2d2d6" },
                new BubbleData { name= "Russia", population= "30916",
color="#09156d" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public class BubbleData
        {
            public string name { get; set; }
            public string population { get; set; }
        }
    }
}

```



Setting the range of the bubble size

The size of the bubbles is calculated from the values got from the `ValuePath` property. The range for the radius of the bubbles can be modified using `MinRadius` and `MaxRadius` properties.

C#HTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.BubbleSettings(bubble =>
    {

bubble.Visible(true).ValuePath("population").MinRadius(20).MaxRadius(40).ValuePath("population").
        DataSource(ViewBag.bubbleData).Add();

    }) .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name") .Add();
    }) .Render();
```

BUBBLESIZE.CS

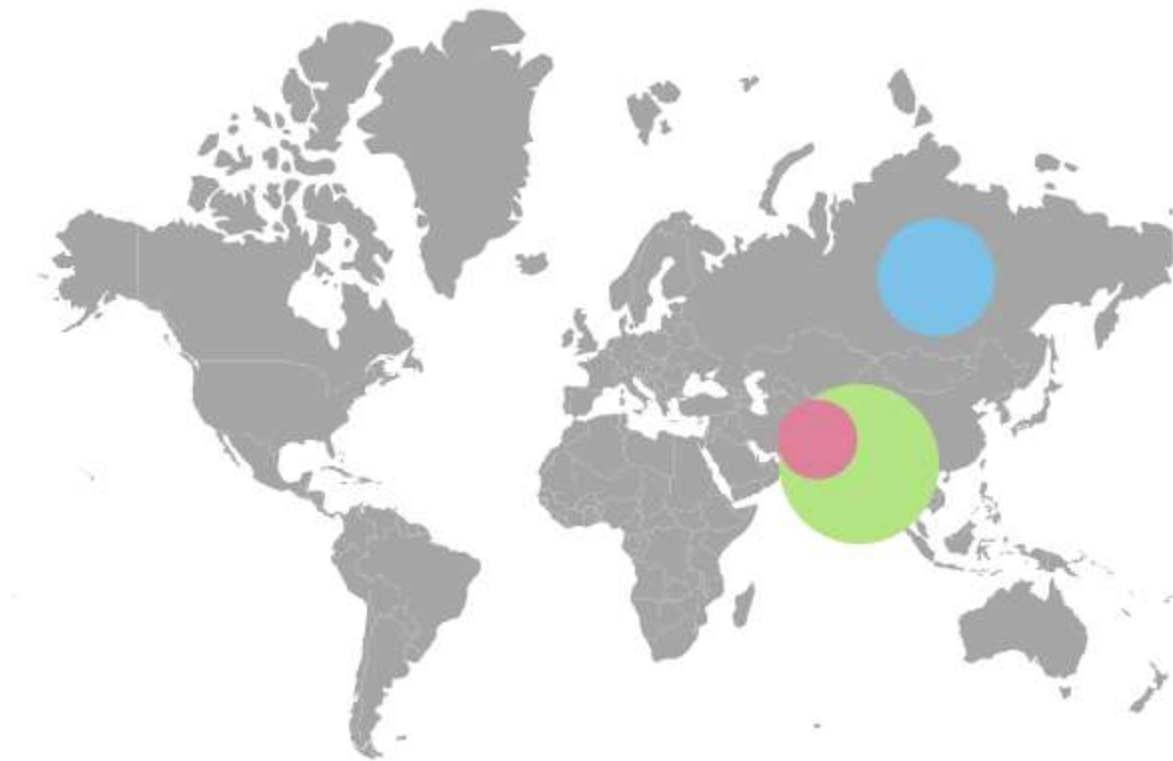
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "India", population= "38332521" },
                new BubbleData { name= "Russia", population= "19651127" },
                new BubbleData { name= "Pakistan", population= "3090416" }
            };
            ViewBag.bubbleData = data;
            return View();
        }

        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }

        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }

        public class BubbleData
        {
            public string name { get; set; }
            public string population { get; set; }
        }
    }
}
```



Multiple bubble groups

Multiple groups of bubbles can be added in the Maps by adding multiple `MapsBubble` in the `MapsBubbles` and customization for the bubbles can be done with the `MapsBubble` class. In the following example, the gender-wise population ratio is demonstrated with two different bubble groups.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    1.BubbleSettings(ViewBag.bubbleSettings).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
}).Render()
```

MULTIPLE-BUBBLE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
```



```

namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { country= "United States",
femaleRatio=50.50442726, maleRatio=49.49557274, femaleRatioColor="green",
maleRatioColor="blue" },
                new BubbleData { country= "India", femaleRatio=48.18032713,
maleRatio=51.81967287, femaleRatioColor="blue", maleRatioColor="#c2d2d6" },
                new BubbleData { country= "Oman", femaleRatio=34.15597234,
maleRatio=65.84402766, femaleRatioColor="#09156d", maleRatioColor="orange"
            },
                new BubbleData { country= "United Arab Emirates",
femaleRatio=27.59638942, maleRatio=72.40361058, femaleRatioColor="#09156d",
maleRatioColor="orange" }
            };
            List<BubbleData> data1 = new List<BubbleData>
            {
                new BubbleData { country= "United States",
femaleRatio=50.50442726, maleRatio=49.49557274, femaleRatioColor="green",
maleRatioColor="blue" },
                new BubbleData { country= "India", femaleRatio=48.18032713,
maleRatio=51.81967287, femaleRatioColor="blue", maleRatioColor="#c2d2d6" },
                new BubbleData { country= "Oman", femaleRatio= 34.15597234,
maleRatio=65.84402766, femaleRatioColor="#09156d", maleRatioColor="orange"
            },
                new BubbleData { country= "United Arab Emirates",
femaleRatio= 27.59638942, maleRatio=72.40361058, femaleRatioColor="#09156d",
maleRatioColor="orange" }
            };
            ViewBag.bubbleData = data;
            ViewBag.bubblesData = data1;
            MapsBubble bubble = new MapsBubble();
            bubble.Visible = true;
            bubble.ValuePath = "femaleRatio";
            bubble.ColorValuePath = "femaleRatioColor";
            bubble.MinRadius = 5;
            bubble.MaxRadius = 20;
            bubble.DataSource = ViewBag.bubbleData;
            MapsBubble bubble1 = new MapsBubble();
            bubble1.Visible = true;
            bubble1.ValuePath = "maleRatio";
            bubble1.ColorValuePath = "maleRatioColor";
            bubble1.MinRadius = 15;
            bubble1.MaxRadius = 25;
            bubble1.DataSource = ViewBag.bubblesData;
            bubble1.BubbleType = Syncfusion.EJ2.Maps.BubbleType.Circle;
            bubble1.Opacity = 0.4;
            List<MapsBubble> bubbleSettings = new List<MapsBubble>();
            bubbleSettings.Add(bubble);
            bubbleSettings.Add(bubble1);
        }
    }
}

```

```
        ViewBag.bubbleSettings = bubbleSettings;
        return View();
    }
    public object GetWorldMap()
    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public class BubbleData
    {
        public string country { get; set; }
        public double femaleRatio { get; set; }
        public double maleRatio { get; set; }
        public string femaleRatioColor { get; set; }
        public string maleRatioColor { get; set; }
    }
}
```



Enable tooltip for bubble

The tooltip for the bubbles can be enabled by setting the `Visible` property of the `MapsTooltipSettings` as `true`. The content for the tooltip can be set using the `ValuePath` property in the `MapsTooltipSettings` of the `MapsBubble` where the value for the `ValuePath` property is the field name from the data source of the `MapsBubble`. Any HTML element can be added as the template in tooltip using the `Template` property.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.BubbleSettings(bubble =>
    {
        bubble.Visible(true).ValuePath("population").MinRadius(20).MaxRadius(40).TooltipSettings(tooltip => tooltip.Visible(true).ValuePath("population")).DataSource(ViewBag.bubbleData).Add();
    })
    .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
}).Render();
```

BUBBLE-TOOLTIP.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "India", population= "38332521" },
                new BubbleData { name= "Australia", population= "383521" },
                new BubbleData { name= "Pakistan", population= "3090416" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/world_map.js");
```

```
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public class BubbleData
    {
        public string name { get; set; }
        public string population { get; set; }
    }
}
```



Legend in ASP.NET MVC Maps Component

A Legend is a visual representation of the symbols used on the Maps. It can be represented in various colors, shapes or other identifiers based on the data and provides valuable information for interpreting what the Maps are displaying. It explains what each symbol in the Maps represents. Legends are enabled by setting the `Visible` property of `MapsLegendSettings` to `true`.

Modes of legend

Legend had two types of mode.

1. **Default** mode
2. **Interactive** mode

Default mode

Default mode legends having symbols with legend labels, used to identify the shape or bubble or marker color. To enable this option by setting the **Mode** property of **MapsLegendSettings** as **Default**.

Interactive mode

The legends can be made interactive with an arrow mark indicating the exact range color in the legend when the mouse hovers over the corresponding shapes. To enable this type of mode by setting the **Mode** property of **MapsLegendSettings** as **Interactive**. The **InvertedPointer** property is used to enable or disable the visibility of the inverted pointer in interactive legend in Maps.

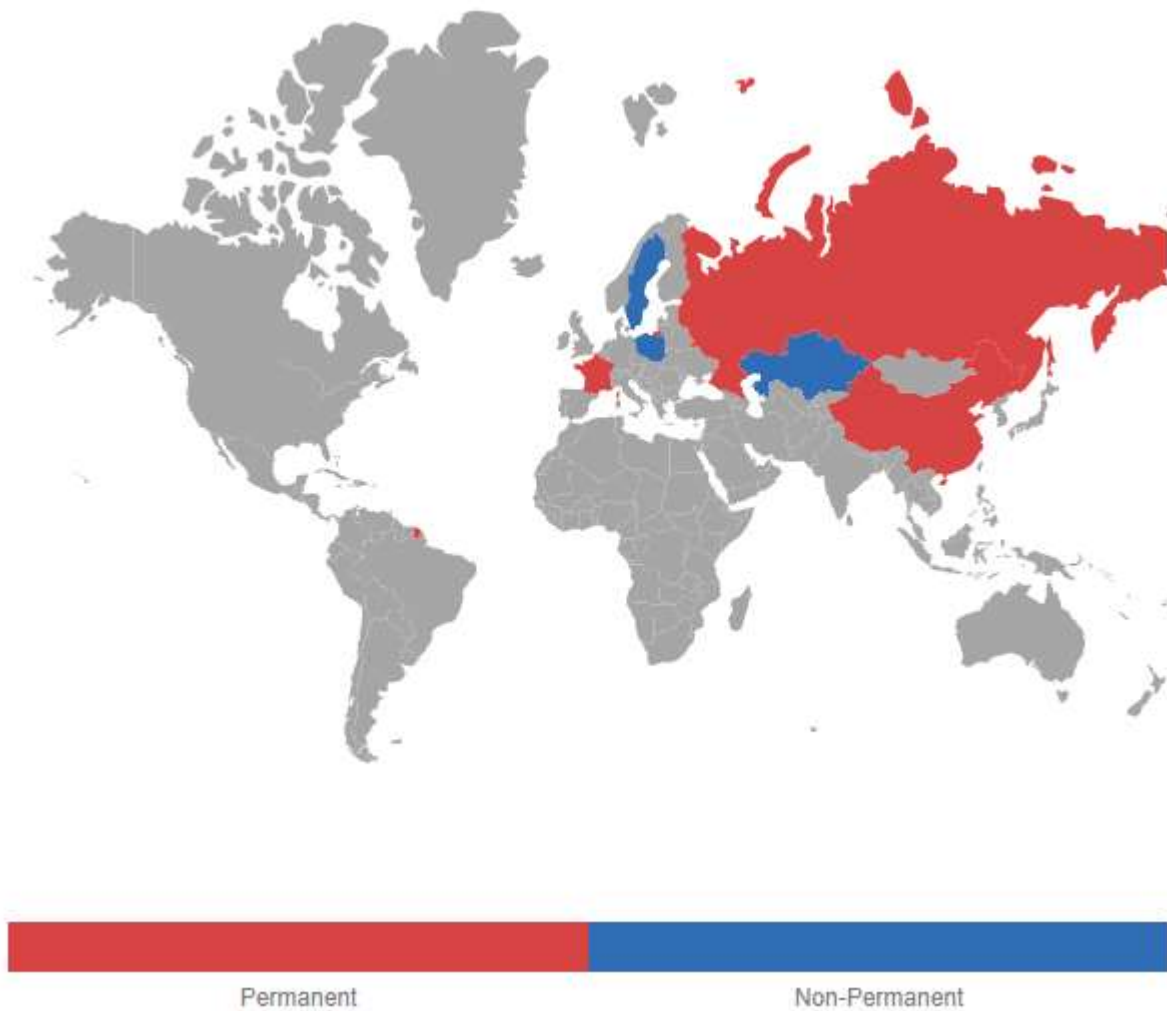
CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new { Country= "China", Membership= "Permanent" },
        new { Country= "France", Membership= "Permanent" },
        new { Country= "Russia", Membership= "Permanent" },
        new { Country= "Kazakhstan", Membership= "Non-Permanent" },
        new { Country= "Poland", Membership= "Non-Permanent" },
        new { Country= "Sweden", Membership= "Non-Permanent" },
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#D84444",Value= "Permanent" },
        new MapsColorMapping { Color= "#316DB5", Value = "Non-Permanent" },
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true).Mode(Syncfusion.EJ2.Maps.LegendMode.Interactive).InvertedPointer(true)).Layers(l =>
{
    l.ShapeSettings(ss =>
ss.ColorValuePath("Membership").ColorMapping(colormapping)).ShapeData(ViewBag.worldMap)
        .ShapeDataPath("Country").ShapePropertyPath("name")
        .DataSource(data).Add();
}).Render()
```

LEGEND-MODE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
```

```
public class HomeController : Controller
{
    public IActionResult Index()
    {
        ViewBag.world_map = GetWorldMap();
        ViewBag.worldMap = GetMap();
        return View();
    }
    public object GetWorldMap()
    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}
```



Positioning of the legend

The legend can be positioned in the following two ways:

- Absolute position
- Dock position

Absolute position

The legend of the Maps can be positioned using the `Location` property in the `MapsLegendSettings`. For positioning the legend based on co-ordinates corresponding to a Maps, the `Position` property is set as **Float**.

Dock position

Legends are positioned in the following locations within the container. The `Position` property in `MapsLegendSettings` is used to set these options in Maps.

- Top
- Left

- Bottom
- Right

The above four positions can be aligned with combination of **Near**, **Center** and **Far** using **Alignment** property in **MapsLegendSettings**. So, the legend can be aligned to 12 positions.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new { Country= "China", Membership= "Permanent" },
        new { Country= "France", Membership= "Permanent" },
        new { Country= "Russia", Membership= "Permanent" },
        new { Country= "Kazakhstan", Membership= "Non-Permanent" },
        new { Country= "Poland", Membership= "Non-Permanent" },
        new { Country= "Sweden", Membership= "Non-Permanent" },
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#D84444",Value= "Permanent" },
        new MapsColorMapping { Color= "#316DB5", Value = "Non-Permanent" },
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true).Position(Syncfusion.EJ2.Maps.LegendPosition.Top).Alignm
ent(Syncfusion.EJ2.Maps.Alignment.Near)).Layers(l =>
{
    l.ShapeSettings(ss =>
ss.ColorValuePath("Membership").ColorMapping(colormapping)).ShapeData(ViewBa
g.worldMap)
        .ShapeDataPath("Country").ShapePropertyPath("name")
        .DataSource(data).Add();
}).Render()
```

LEGEND-POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
    }
}
```



```
public object GetWorldMap()  
{  
    string allText =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");  
    return JsonConvert.DeserializeObject(allText);  
}  
public object GetMap()  
{  
    string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
    return JsonConvert.DeserializeObject(allText, typeof(object));  
}  
}
```

● Permanent ● Non-Permanent



Legend for shapes

Legend for shapes can be generated from color mapping types such as equal color mapping, range color mapping and desaturation color mapping.

The below code snippet demonstrate the equal color mapping legends for the shapes. To bind the given data source to the `DataSource` property of `MapsLayer`. Set the value of `ShapePropertyPath` to `name` and `ShapeDataPath` to `Country`. To enable equal color mapping, set the multiple `MapsColorMapping` to the `MapsShapeSettings`. Finally, set the `Visible` property of `MapsLegendSettings` as `true`. The `Label` property in `MapsColorMapping` is used to set the text name for legend in Maps.

CSHTML

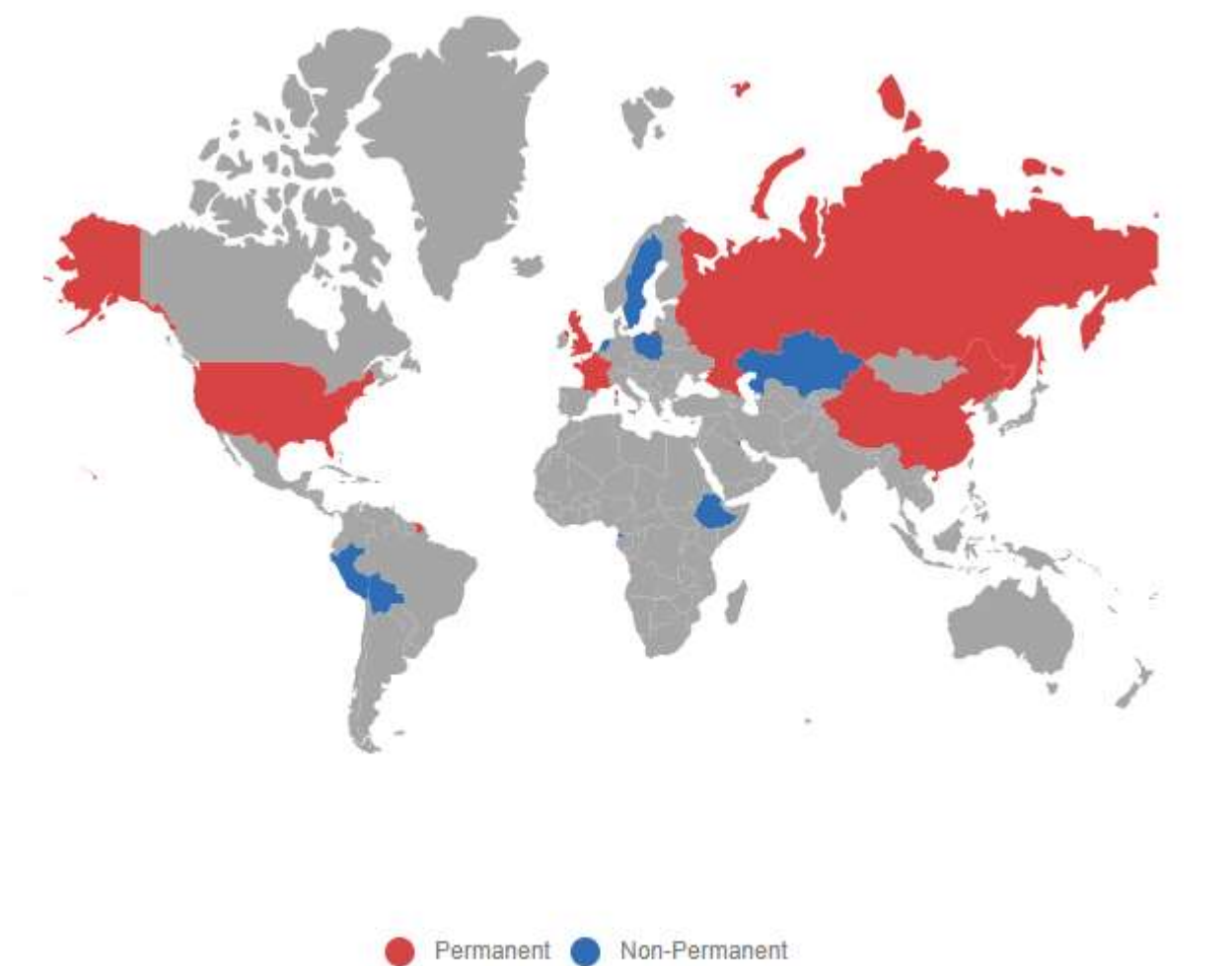
```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var ColorMapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping>
    {
        new MapsColorMapping{Value="Permanent", Color="#D84444"},
        new MapsColorMapping{Value="Non-Permanent", Color="#316DB5"}
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true)).Layers(l =>
{
    l.ShapeSettings(ss =>
ss.ColorValuePath("Membership").ColorMapping(ColorMapping))

    .ShapeData(ViewBag.worldMap).ShapeDataPath("Country").ShapePropertyPath("name")
        .DataSource(ViewBag.uncountries).Add();
}).Render()
```

LEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Maps;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            ViewBag.unCountries = GetElectionData();
            ViewBag.uncountries = GetData();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
```

```
        return JsonConvert.DeserializeObject(text);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetData()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/electiondata.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetElectionData()
    {
        string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/electiondata.json");
        return JsonConvert.DeserializeObject(text);
    }
}
```



Legend shape

Maps supports the following types of legend shapes. The `Shape` property in the `MapsLegendSettings` class can be used to change the type of legend shapes.

- Circle
- Rectangle
- Triangle
- Diamond
- Cross
- Star
- HorizontalLine
- VerticalLine
- Pentagon
- InvertedTriangle

The shape of legends can be customized by using the `ShapeWidth`, `ShapeHeight`, `ShapeBorder` and `ShapePadding` properties.

Customization

The following properties are available in legend to customize the legend shape and legend text in Maps.

- **Background** - To customize the background color of the Legend.
- **Border** - To customize the color, width and opacity of the border for the Legend.
- **Fill** - To apply the color for the Legend.
- **LabelDisplayMode** - To customize the display mode for the Legend text.
- **LabelPosition** - To customize the position of the Legend text.
- **Orientation** - To customize the orientation of the Legend.
- **TextStyle** - To customize the text style for Legend.
- **Title** - To apply the title for the Legend.
- **TitleStyle** - To customize the style of the title for the Legend.
- **Height** - To customize the height of the Legend.
- **Width** - To customize the width of the Legend.
- **Opacity** - To apply the opacity to the Legend.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new { Country= "China", Membership= "Permanent" },
        new { Country= "France", Membership= "Permanent" },
        new { Country= "Russia", Membership= "Permanent" },
        new { Country= "Kazakhstan", Membership= "Non-Permanent" },
        new { Country= "Poland", Membership= "Non-Permanent" },
        new { Country= "Sweden", Membership= "Non-Permanent" },
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#D84444",Value= "Permanent" },
        new MapsColorMapping { Color= "#316DB5", Value = "Non-Permanent" },
    };
    var text = new MapsFont
    {
        Size = "12px",
        Color = "red",
        FontStyle = "italic"
    };
    var title = new MapsCommonTitleSettings
    {
        Description = "Legend title",
        Text = "Legend"
    };
    var titleStyle = new TitleSettingsTextStyleTitleSettings
    {
        Size = "12px",
        Color = "#d6e341",
        FontStyle = "italic"
    };
    var border = new MapsBorder
    {

```

```

        Color = "blue",
        Width = 2,
        Opacity = 1
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true).Background("green").Fill("orange").LabelPosition(Syncfu
sion.EJ2.Maps.LabelPosition.Before).
Orientation(Syncfusion.EJ2.Maps.LegendArrangement.Vertical).TextStyle(text).
Title(title).TitleStyle(titleStyle).Border(border)).Layers(l =>
{
    l.ShapeSettings(ss =>
ss.ColorValuePath("Membership").ColorMapping(colormapping)).ShapeData(ViewBa
g.worldMap)
        .ShapeDataPath("Country").ShapePropertyPath("name")
        .DataSource(data).Add();
}).Render()

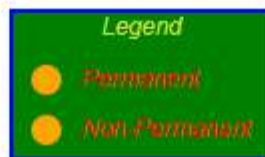
```

LEGEND-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Legend for items excluded from color mapping

The legend can be enabled for items excluded from the color mapping using the `Color` property in `MapsColorMapping`. Refer to the **population_density** data which demonstrates the population density of some countries.

```
`sh
[
...
{
'code': 'AE',
'value': 90,
'name': 'United Arab Emirates',
'population': 8264070,
'density': 99
```

```

},
{
  'code': 'GB',
  'value': 257,
  'name': 'United Kingdom',
  'population': 62041708,
  'density': 255
},
{
  'code': 'US',
  'value': 34,
  'name': 'United States',
  'population': 325020000,
  'density': 33
}
...
];
`

```

In the following example, color mapping is added for the ranges from **0** to **200**. If there are any records in the data source that are outside of this range, the color mapping will not be applied. To apply the color for these excluded items, set the **Color** property alone in the **MapsColorMapping**. To enable legend for these items, set the **Visible** property of **MapsLegendSettings** to **true**.

CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var colorMapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping>
    {
        new MapsColorMapping {From = 1, To = 100, Color =
"rgb(153,174,214)"},
        new MapsColorMapping {From = 101, To = 200, Color =
"rgb(115,143,199)"},
        new MapsColorMapping {Color = "rgb(77,112,184)"}
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true)).Layers(l =>
{
    l.ShapeSettings(ss =>
ss.ColorValuePath("density").ColorMapping(colorMapping)).ShapeData(ViewBag.w
orldMap).ShapeDataPath("name").ShapePropertyPath("name")
.DataSource(ViewBag.populationDensity).Add();
}
)

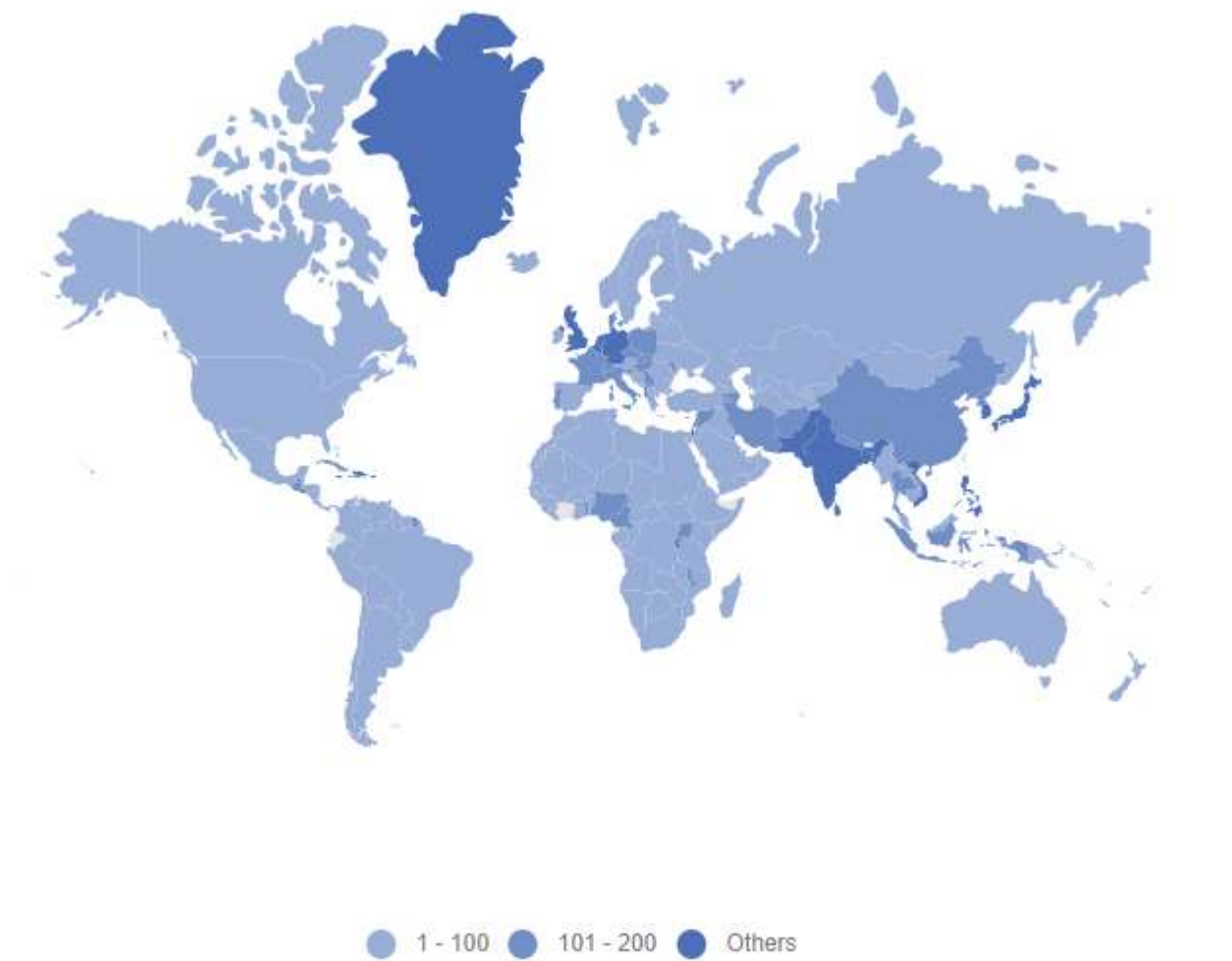
```



```
}).Render()
```

EXCLUDELEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.worldMap = GetMap();
            ViewBag.populationDensity = GetPopulationDensity();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetPopulationData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/populationdensity.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetPopulationDensity()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/populationdensity.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Hide desired legend items

Use the **ShowLegend** property in the **MapsColorMapping** to show or hide the desired legend items in Maps. If the **ShowLegend** property is set to **false**, the legend item will be hidden. otherwise, it will be visible.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var colorMapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping {From = 1, To = 100, Color =
"rgb(153,174,214)", ShowLegend=true},
        new MapsColorMapping {From = 100, To = 200, Color =
"rgb(115,143,199)", ShowLegend=true},
        new MapsColorMapping {Color = "rgb(77,112,184)", ShowLegend=false},
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true)).Layers(l =>
{
```

```

        l.ShapeSettings(ss =>
            ss.ColorValuePath("density").ColorMapping(colorMapping))

        .ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name")
            .DataSource(ViewBag.populationDensity).Add();
    }).Render()

```

HIDELEGEND.CS

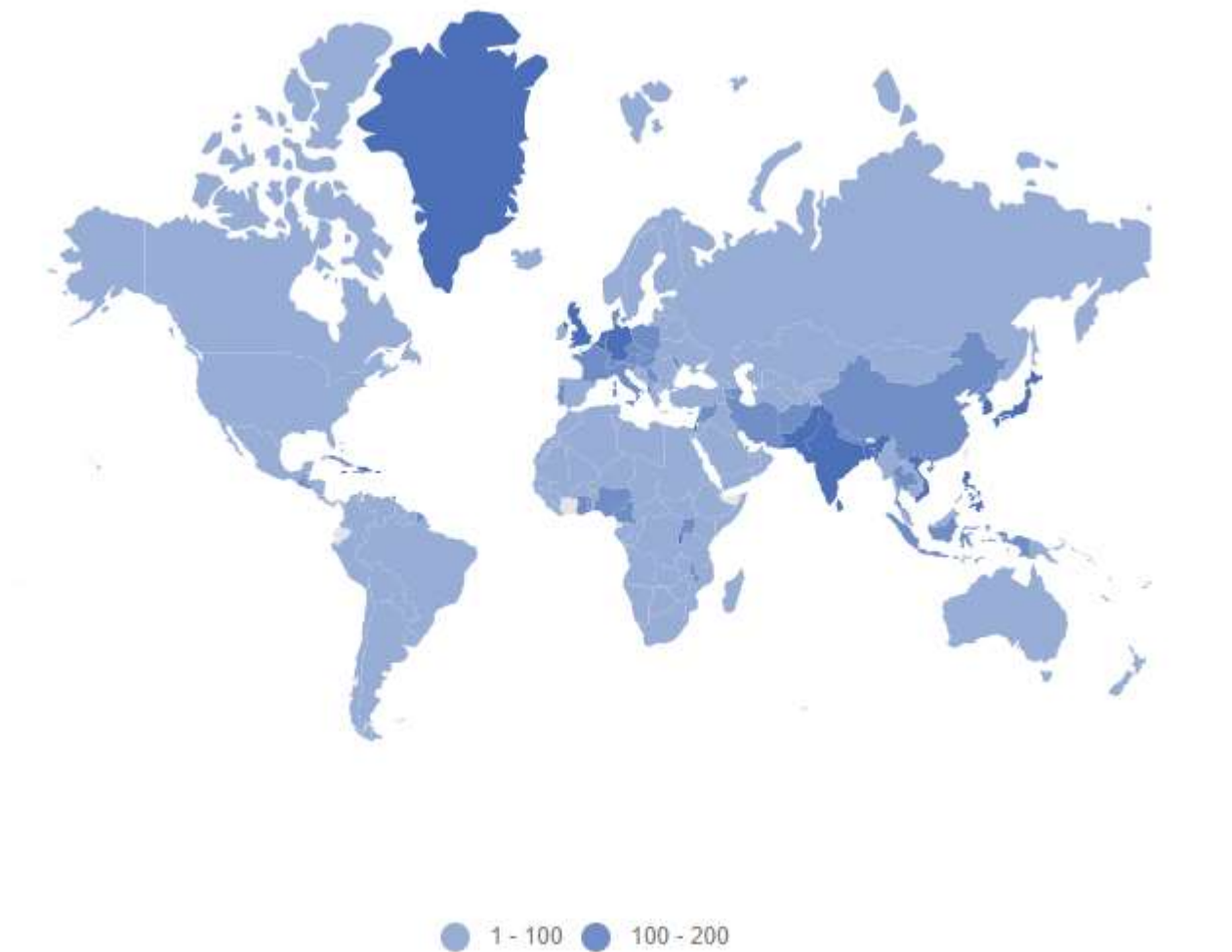
```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.worldMap = GetMap();
            ViewBag.populationDensity = GetPopulationDensity();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
                System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetPopulationData()
        {
            string text =
                System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/populationdensity.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
                System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetPopulationDensity()
        {
            string allText =
                System.IO.File.ReadAllText(Server.MapPath("~/App_Data/populationdensity.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

```
}

```



Hide legend items based on data source value

Depending on the boolean values provided in the data source, the legend items will be hidden or visible. Bind the field name that contains the visibility state in the data source to the **ShowLegendPath** property of the **MapsLegendSettings** class to achieve this.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true).ShowLegendPath("visibility").Layers(l =>
{
    l.ShapeSettings(ss =>
ss.ColorValuePath("color")).ShapeData(ViewBag.worldMap)
        .ShapeDataPath("continent").ShapePropertyPath("continent")
        .DataSource(ViewBag.populationDensity).Add();
    }) .Render()
})
```

HIDELEGENDBASEDDS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.worldMap = GetMap();
            ViewBag.populationDensity = GetPopulationDensity();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetPopulationData()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/populationdensity.js
n");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetPopulationDensity()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/populationdensity.js
n"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Binding legend item text from data source

To show the legend text based on values provided in the data source, use the **ValuePath** property in the **MapsLegendSettings**.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").LegendSettings(legend =>
    legend.Visible(true).ValuePath("continent")).Layers(1 =>
    {
        1.ShapeSettings(ss =>
            ss.ColorValuePath("color")).ShapeData(ViewBag.worldMap)
            .ShapeDataPath("continent").ShapePropertyPath("continent")
            .DataSource(ViewBag.legendData1).Add();
    }).Render()
```

BINDLEGENDTEXT.CS

```
using System;
```

```
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.legendData = GetLegendData();
            ViewBag.legendData1 = GetLegendData1();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetLegendData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/LegendData.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetLegendData1()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/LegendData.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Hide duplicate legend items

To hide the duplicate legend items in Maps, set the `RemoveDuplicateLegend` property to **true** in the `MapsLegendSettings`.

CSHTML

```
@using Syncfusion.EJ2;  
@using Syncfusion.EJ2.Maps;  
@Html.EJS().Maps("maps").LegendSettings(legend =>  
    legend.Visible(true).ValuePath("continent")  
    .RemoveDuplicateLegend(true)).Layers(l =>  
    {  
        l.ShapeSettings(ss =>  
            ss.ColorValuePath("color")).ShapeData(ViewBag.worldMap)  
            .ShapeDataPath("continent").ShapePropertyPath("continent")  
            .DataSource(ViewBag.legendData1).Add();  
    }).Render()
```

DUPLICATELEGEND.CS


```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.legendData = GetLegendData();
            ViewBag.legendData1 = GetLegendData1();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetPopulationData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/LegendData.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetLegendData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/populationdensity.js"
);
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public object GetLegendData1()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/LegendData.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Toggle option in legend

The toggle option has been provided for legend. If the legend can be toggled, the given color will be changed to the corresponding Maps shape item. To enable the toggle options in Legend, set the **Enable** property of the **MapsToggleLegendSettings** to **true**.

The following properties are available to customize the toggle option in legend.

- **ApplyShapeSettings** – To apply the **Fill** property value to the shape of the Maps when toggling the legend items.
- **Fill** - To apply the color to the shape of the Maps for which legend item is toggled.
- **Opacity** – To customize the transparency for the shapes for which legend item is toggled.
- **Border** – To customize the color, width and opacity of the border of the shapes in Maps.

CSHTML

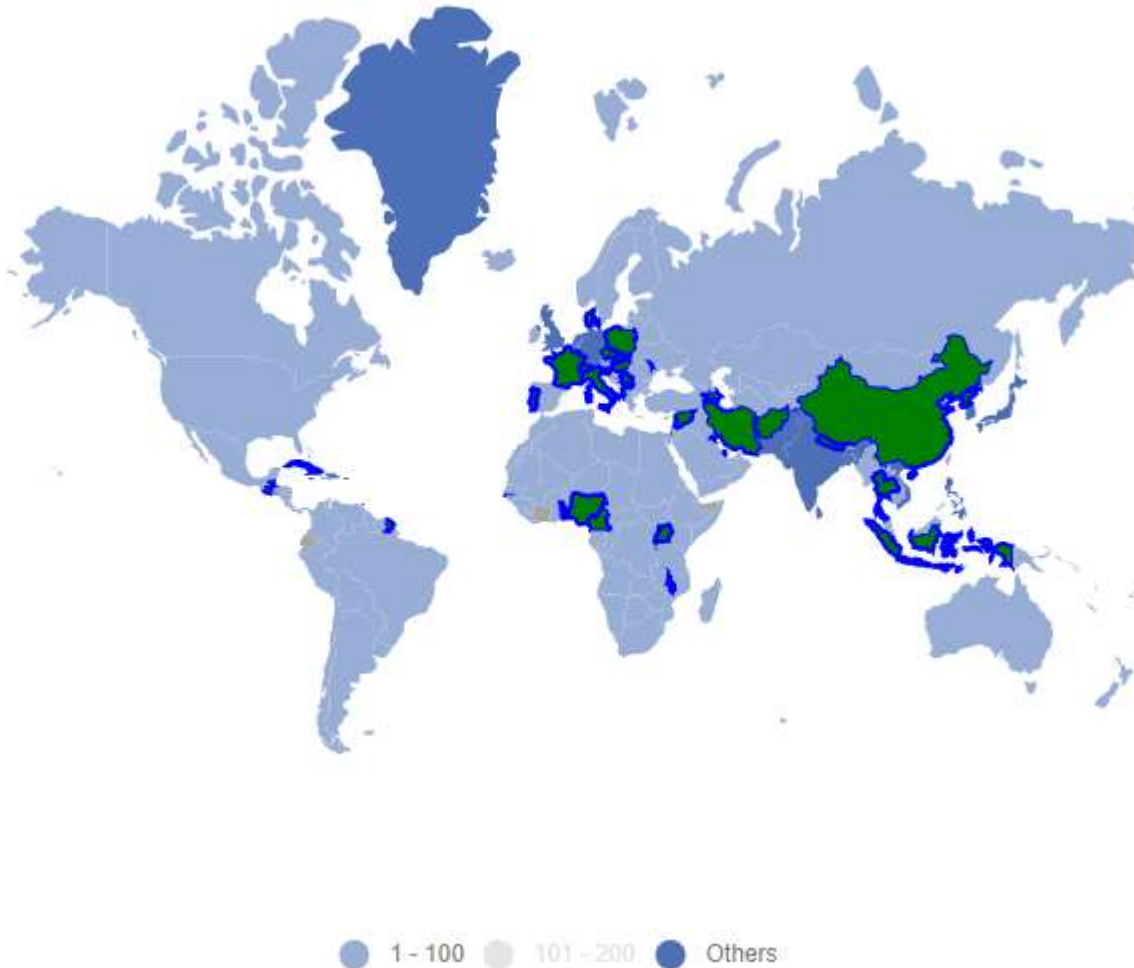
```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("container").LegendSettings(legend =>
    legend.Visible(true).ToggleLegendSettings(Tl => Tl.Enable(true))
```

```
.ApplyShapeSettings(false).Fill("green").Border(Br =>
Br.Color("green").Width(2).Opacity(1))).Layers(layer =>
{
    layer.DataSource(ViewBag.populationDensity).ShapeDataPath("name")
    .ShapePropertyPath("name").ShapeSettings(new MapsShapeSettings
    {
        ColorValuePath = "density",
        ColorMapping = new List<MapsColorMapping> {
            new MapsColorMapping { From = 1 , To = 100,
Color="rgb(153,174,214)" },
            new MapsColorMapping { From = 101 , To = 200,
Color="rgb(115,143,199)" },
            new MapsColorMapping { Color="rgb(77,112,184)" },
        }
    }).ShapeData(ViewBag.worldMap).Add();
}).Render()
```

TOGGLELEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.populationData = GetPopulationData();
            ViewBag.worldMap = GetMap();
            ViewBag.populationDensity = GetPopulationDensity();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetPopulationData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/populationdensity.js
n");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetPopulationData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/populationdensity.js
n");
```

```
        return JsonConvert.DeserializeObject(text);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object GetPopulationDensity()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/populationdensity.json
"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}
```



Enable legend for bubbles

To enable the legend for the bubble by setting the **Visible** property of **MapsLegendSettings** as **true** and **Type** property of **MapsLegendSettings** as **Bubbles**.

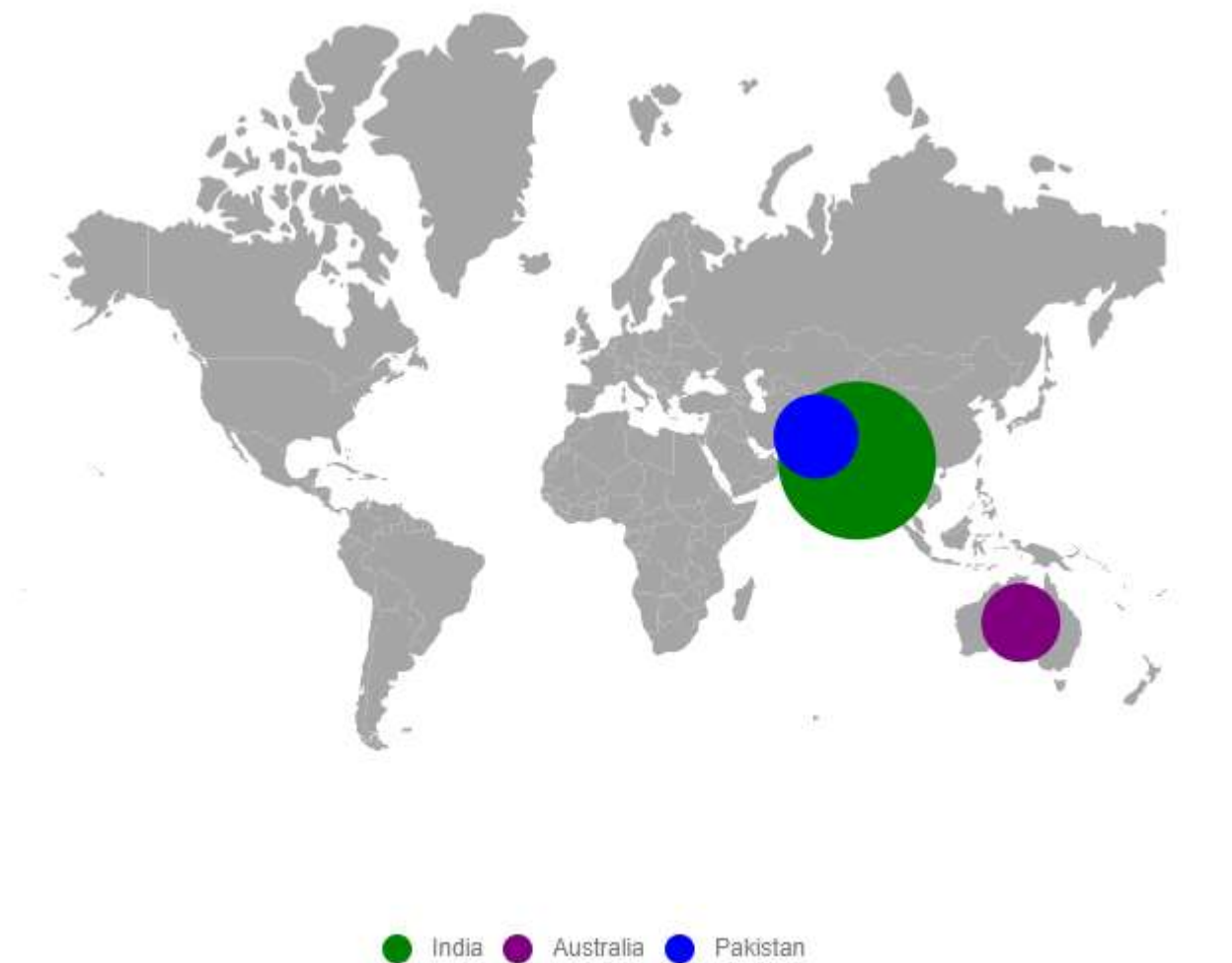
CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true).Type(LegendType.Bubbles)).Layers(l =>
{
    l.BubbleSettings(bubble =>
    {
        bubble.Visible(true).MinRadius(20).MaxRadius(40).ColorValuePath("color").ValuePath("population").DataSource(ViewBag.bubbleData).Add();
    }).ShapeData(ViewBag.worldMap).ShapeDataPath("name").ShapePropertyPath("name").Add();
}).Render()
```

BUBBLE-LEGEND.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "India", population= "38332521",
color="green" },
                new BubbleData { name= "Australia", population= "383521",
color="purple" },
                new BubbleData { name= "Pakistan", population= "3090416",
color="blue" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```

```
}  
public object GetMap()  
{  
    string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
    return JsonConvert.DeserializeObject(allText, typeof(object));  
}  
public class BubbleData  
{  
    public string name { get; set; }  
    public string population { get; set; }  
    public string color { get; set; }  
}  
}
```



Enable legend for markers

To enable legend for marker by setting the **Visible** property of **MapsLegendSettings** as **true** and **Type** property of **MapsLegendSettings** as **Markers**. The **LegendText** property in the **MapsMarker** can be used to show the legend text based on values provided in the data source.

CHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new {latitude= 37.0000, longitude= -120.0000, city= "California"
    },
        new {latitude= 40.7127, longitude= -74.0059, city= "New York" },
        new {latitude= 42.0000, longitude= -93.0000, city= "Iowa" }
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true).Type(LegendType.Markers)).Layers(l =>
{
    l.MarkerSettings(marker =>
    {
        marker.Visible(true).Shape(MarkerType.Diamond).LegendText("city").DataSource
        (data).Add();
        }).ShapeData(ViewBag.worldmap).Add();
    }).Render()

```

LEGEND.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            List<MarkerData> data = new List<MarkerData>
            {
                new MarkerData {latitude= 37.0000, longitude= -120.0000,
city= "California" },
                new MarkerData {latitude= 40.7127, longitude= -74.0059,
city= "New York" },
                new MarkerData {latitude= 42.0000, longitude= -93.0000,
city= "Iowa" }
            };
            ViewBag.markerData = data;
            return View();
        }
        public object GetWorldMap()
        {

```

```
string allText =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");  
return JsonConvert.DeserializeObject(allText);  
}  
public class MarkerData  
{  
    public double latitude { get; set; }  
    public double longitude { get; set; }  
    public string city { get; set; }  
}  
}
```



● California ● New York ● Iowa

Imitate/Map marker shape to the legend shape

To imitate or map the marker shape with its legend item shape, set the `UseMarkerShape` property to **true** in the `MapsLegendSettings` property.

CSHTML


```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("container").LegendSettings(legend =>
legend.Visible(true).Type(Syncfusion.EJ2.Maps.LegendType.Markers).UseMarkersS
hape(true).ToggleLegendSettings(toggle=>
toggle.ApplyShapeSettings(false))).Layers(l =>
{
    l.MarkerSettings(marker =>
    {

marker.Visible(true).ColorValuePath("color").ShapeValuePath("shape").LegendT
ext("Country").DataSource(ViewBag.markerdata).Add();

}).ShapeSettings(ss=>ss.Fill("#E5E5E5")).ShapeData(ViewBag.world_map).Add();
}).Render()

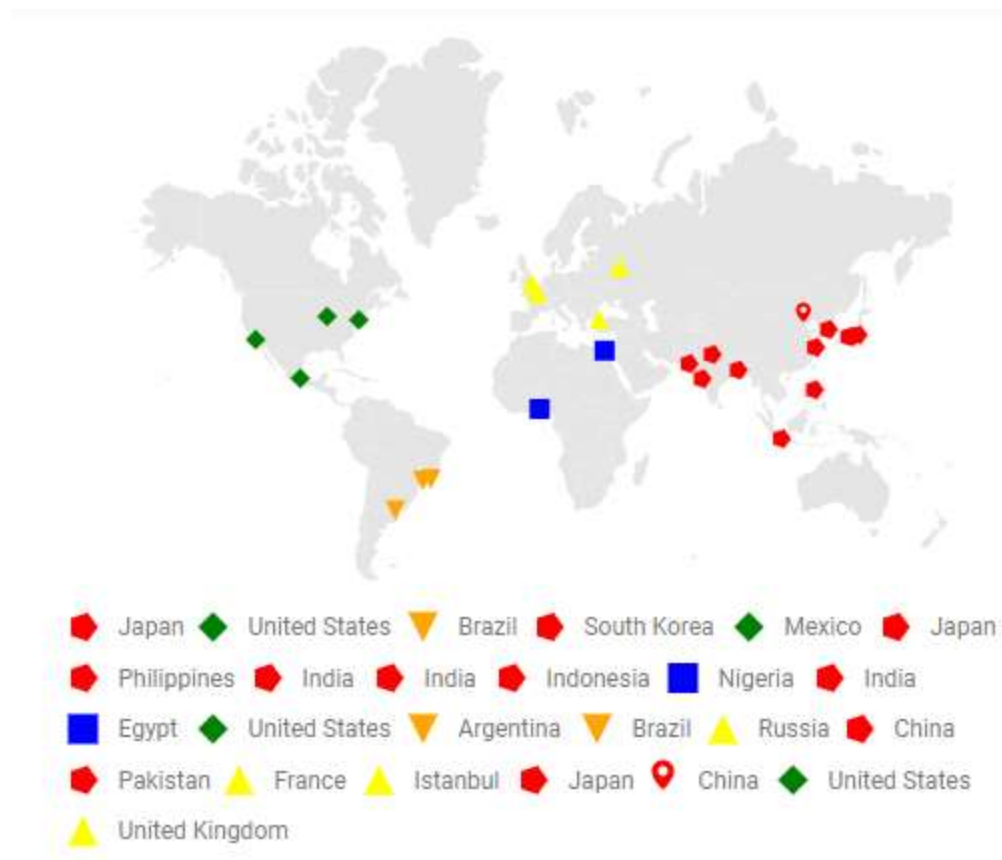
```

MARKER-LEGEND.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.markerdata = GetMarkerData();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMarkerData()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/markerdata.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}

```



Annotations in ASP.NET MVC Maps Component

<!-- markdownlint-disable MD013 -->

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Annotation

By using the **Content** property of **MapsAnnotation**, text content or id of an element or an HTML string can be specified to render a new HTML element in Maps.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Annotations(new
List<Syncfusion.EJ2.Maps.MapsAnnotation>{
    new Syncfusion.EJ2.Maps.MapsAnnotation
    {
        Content = "<div id='annotation' style='display:none'><img
src='~/App_Data/ballon.png'></div>",
        X = "0%",
        Y = "50%",
    }
}).Layers(layer =>
{
    layer.ShapeData(ViewBag.worldmap).Add();
}).Render()
```

ANNOTATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



Annotation customization

Changing the z-index

The stack order of an annotation element can be changed using the **ZIndex** property in the **MapsAnnotation**.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Annotations(new
List<Syncfusion.EJ2.Maps.MapsAnnotation>{
    new Syncfusion.EJ2.Maps.MapsAnnotation
    {
        Content = "<div id='first'><h1>Maps</h1></div>",
        X = "20%",
        Y = "50%",
        ZIndex = "-1"
    }
}).Layers(layer =>
{
    layer.ShapeData(ViewBag.worldmap).Add();
}).Render()
```

ANNOTATION-ZINDEX.CS

```
using System;
using System.Collections.Generic;
```

```
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



Positioning an annotation

Annotations can be placed anywhere in the Maps by specifying pixel or percentage values to the **X** and **Y** properties in the **MapsAnnotation**.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Annotations(new
List<Syncfusion.EJ2.Maps.MapsAnnotation>{
    new Syncfusion.EJ2.Maps.MapsAnnotation
    {
        Content = "<div id='first'><h1>Maps</h1></div>",
        X = "40%",
        Y = "50%",
        ZIndex = "-1"
    }
}).Layers(layer =>
{
    layer.ShapeData(ViewBag.worldmap).Add();
}).Render()
```

ANNOTATION-POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



Alignment of an annotation

Annotations can be aligned using the `HorizontalAlignment` and `VerticalAlignment` properties in the `MapsAnnotation`. The possible values can be **Center**, **Far**, **Near** and **None**.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Annotations(new
List<Syncfusion.EJ2.Maps.MapsAnnotation>{
    new Syncfusion.EJ2.Maps.MapsAnnotation
    {
        Content = "<div id='first'><h1>Maps</h1></div>",
        X = "20%",
        Y = "50%",
        ZIndex = "-1",
        HorizontalAlignment = AnnotationAlignment.Center
    }
}).Layers(layer =>
{
    layer.ShapeData(ViewBag.worldmap).Add();
}).Render()
```

ANNOTATION-ALIGNMENT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
```

```
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
                System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



Multiple Annotation

Multiple annotations can be added to the Maps by adding Multiple `MapsAnnotation` in the `MapsAnnotations` and customization for the annotations can be done with the `MapsAnnotation`.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
```



```

@Html.EJS().Maps("maps").Annotations(new
List<Syncfusion.EJ2.Maps.MapsAnnotation>{
    new Syncfusion.EJ2.Maps.MapsAnnotation
    {
        Content = "<div id='first'><h1>Maps</h1></div>",
        X = "50%",
        Y = "0%",
        ZIndex = "-1"
    },
    new Syncfusion.EJ2.Maps.MapsAnnotation {
        Content = "<div id='first'><h1>Maps-Annotation</h1></div>",
        X = "20%",
        Y = "50%",
        ZIndex = "-1"
    }
}).Layers(layer =>
{
    layer.ShapeData(ViewBag.worldmap).Add();
}).Render()

```

MULTIPLE-ANNOTATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}

```



Navigation lines

The navigation lines are used to denote the path between two locations. This feature can be used to draw flight or sea routes. Navigation lines are enabled by setting the `Visible` property of the `MapsNavigationLine` to `true`.

Customization

The following properties are available in `MapsNavigationLine` to customize the navigation line of the Maps component.

- `Color` - To apply the color for navigation lines in Maps.
- `DashArray` - To define the pattern of dashes and gaps that is applied to the outline of the navigation lines.
- `Width` - To customize the width of the navigation lines.
- `Angle` - To customize the angle of the navigation lines.
- `HighlightSettings` - To customize the highlight settings of the navigation line.
- `SelectionSettings` - To customize the selection settings of the navigation line.

To navigate the line between two cities on the world map, `Latitude` and `Longitude` values are used to indicate the start and end points of navigation lines drawn on Maps.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").LegendSettings(legend=>
    legend.Visible(true)).Layers(l =>
    {
        l.NavigationLineSettings(ns => {
```

```
        ns.Visible(true).Latitude(new double[] { 40.7128, 36.7783})
        .Longitude(new double[] { -74.0060, -119.4179
    }).Color("black").Angle(90)
        .Width(4).DashArray("4").Add();
    }).ShapeData(ViewBag.worldmap).Add();
}).Render()
```

NAVIGATION-LINE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



Enabling the arrows

To enable the arrow in the navigation line, set the **ShowArrow** property of **MapsArrow** to **true**. The following properties are available in **MapsArrow** to customize the arrow of the navigation lines.

- **Color** - To apply the color for arrow of the navigation line.
- **Offset** - To customize the offset position of the arrow of the navigation line.
- **Position** - To customize the position of the arrow in navigation line. The possible values can be **Start** and **End**.
- **Size** - To customize the size of the arrow in pixels.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").LegendSettings(legend =>
    legend.Visible(true)).Layers(l =>
    {
        l.NavigationLineSettings(ns =>
            {
                ns.Visible(true).Latitude(new double[] { 40.7128, 36.7783 })
                .Longitude(new double[] { -74.0060, -119.4179
            }).Color("black").Angle(90)
                .Width(4).DashArray("4").ArrowSettings(new
                Syncfusion.EJ2.Maps.MapsArrow
                {
                    ShowArrow = true,
                    Size = 15,
```

```
        Position = "Start",
    }).Add();
    }).ShapeData(ViewBag.worldmap).Add();
}).Render()
```

ARROW.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```



User Interactions

Zooming

The zooming feature is used to zoom in and out of Maps to show in-depth information. It is controlled by the `ZoomFactor` property of the `MapsZoomSettings` of the Maps. The `ZoomFactor` is increased or decrease dynamically based on zoom in and out interaction.

Enable zooming

Zooming of Maps is enabled by setting the `Enable` property of `MapsZoomSettings` to **true**.

Enable panning

To enable the panning feature, set the `EnablePanning` property of `MapsZoomSettings` to **true**.

CSHTML

```
@using Syncfusion.EJ2;  
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true).EnablePanning(  
true)).Layers(l=> {  
    l.ShapeData(ViewBag.worldMap).Add();  
}).Render()
```

PANNING.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;
```

```

using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

Various type of zooming

Zooming can be performed in following types:

Zooming toolbar

By default, the toolbar is rendered with **zoom-in**, **zoom-out**, and **reset** options when it is set to **true** in the **Enable** property of **MapsZoomSettings**.

The following options are available in toolbar.

1. Zoom - Provides rectangular zoom support.
2. ZoomIn - Zoom in the Maps.
3. ZoomOut - Zoom out the Maps.
4. Pan - Switches to panning if rectangular zoom is activated.
5. Reset - Restores the Maps to the default view.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Load("load").ZoomSettings(zoom =>
zoom.Enable(true)).Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()
<script>
    function load(args) {

```

```

var maps = document.getElementById('maps').ej2_instances[0];
maps.zoomSettings.toolbarSettings.buttonSettings.toolbarItems =
['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'];
}
</script>

```

ZOOM.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

Pinch zooming

To enable or disable the pinch zooming, use the `PinchZooming` property in `MapsZoomSettings`.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true).PinchZooming(true)).Layers(l=> {
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

PINCHZOOM.CS

```

using System;

```



```

using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

Single-click zooming

To enable or disable the single-click zooming, use the `ZoomOnClick` property in `MapsZoomSettings`.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true).ZoomOnClick(true)).Layers(l=> {
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

SINGLECLICK.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller

```

```

{
    public IActionResult Index()
    {
        ViewBag.world_map = GetWorldMap();
        ViewBag.worldMap = GetMap();
        return View();
    }
    public object GetWorldMap()
    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}
}

```

Double-click zooming

To enable or disable the double-click zooming, use the `DoubleClickZoom` property in `MapsZoomSettings`.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true).DoubleClickZoom(true)).Layers(l=> {
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

DOUBLECLICK.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()

```

```

    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```

Mouse wheel zooming

To enable or disable mouse wheel zooming, use the `MouseWheelZoom` property in `MapsZoomSettings`.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true).MouseWheelZoom
(true)).Layers(l=> {
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

MOUSEWHEEL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {

```

```

        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```

Selection zooming

To enable or disable selection zooming, use the `EnableSelectionZooming` property in `MapsZoomSettings`. The `EnablePanning` property must be set to **false** to enable the selection zooming in Maps.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom =>
zoom.Enable(true).EnableSelectionZooming(true).EnablePanning(false)).Layers(
1 =>
{
    1.ShapeData(ViewBag.worldMap).Add();
}).Render()

```

SELECTIONZOOMING.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

```
}
```

Setting minimum and maximum values for zoom factor

The zooming range can be adjusted using the `MinZoom` and `MaxZoom` properties in `MapsZoomSettings`. The `minZoom` value is set to 1 by default, and the `maxZoom` value is set to 10.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom =>
zoom.Enable(true).MinZoom(2).MaxZoom(12)).Layers(l =>
{
    l.ShapeData(ViewBag.worldMap).Add();
}).Render()
```

MINZOOMING.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```

Zooming with animation

To zoom in or zoom out the Maps with animation, use the `AnimationDuration` property in `MapsLayer`.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").ZoomSettings(zoom=>zoom.Enable(true)).Layers(l=> {
    l.AnimationDuration(500).ShapeData(ViewBag.worldMap).Add();
}).Render();
```

ANIMATIONZOOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```

Customizing the zoom toolbar

The zoom toolbar can be customized by using the **ToolbarSettings** option in the [ZoomSettings](#). The following properties can be used to customize the zoom toolbar.

- **BackgroundColor** - It is used to customize the background color of the zoom toolbar.
- **BorderOpacity** - It is used to customize the opacity of the border of the zoom toolbar.
- **BorderWidth** - It is used to customize the thickness of the border of the zoom toolbar.
- **BorderColor** - It is used to customize the color of the border of the zoom toolbar.
- **HorizontalAlignment** - It is used to position the zoom toolbar in near, far, and center positions to customize its horizontal placement.
- **VerticalAlignment** - It is used to position the zoom toolbar in near, far, and center positions to customize its vertical placement.

- **Orientation** - It is used to change the orientation (horizontal/vertical) of the zoom toolbar.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Load("load").ZoomSettings(zoom =>
zoom.Enable(true)).Layers(l =>
{

l.ShapeSettings(sh=>sh.Fill("#C1DFF5")).ShapeData(ViewBag.worldMap).Add();
}).Render()
<script>
    function load(args) {
        var maps = document.getElementById('maps').ej2_instances[0];
        maps.zoomSettings.toolbarSettings.orientation = "Vertical";
        maps.zoomSettings.toolbarSettings.backgroundColor = "pink";
        maps.zoomSettings.toolbarSettings.borderWidth = 3;
        maps.zoomSettings.toolbarSettings.borderColor = "green";
        maps.zoomSettings.toolbarSettings.verticalAlignment = "Near";
        maps.zoomSettings.toolbarSettings.buttonSettings.toolbarItems =
        ['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'];
    }
</script>
```

ZOOMTOOLBAR.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```

```
}  
}
```



<!-- markdownlint-disable MD036 -->

Customizing the buttons in the zoom toolbar

The appearance of the buttons in the zoom toolbar can be customized by using the `ButtonSettings` option in the `ToolbarSettings` of the [ZoomSettings](#) property. The following properties can be used to customize the zoom toolbar buttons.

- `Fill` - It is used to set the background color of the buttons.
- `Color` - It is used to customize the color of the icons inside the button.
- `BorderOpacity` - It is used to set the opacity of the border of the zoom toolbar buttons.
- `BorderWidth` - It is used to set the thickness of the border of the zoom toolbar buttons.
- `BorderColor` - It is used to set the color of the border of the zoom toolbar buttons.
- `Radius` - It is used to set the size of the button.
- `SelectionColor` - It is used to set the color of the icons inside the button when selection is performed.
- `HighlightColor` - It is used to change the color of the button when the mouse is hovered over it.
- `Padding` - It is used to change the padding space between each button.
- `Opacity` - It is used to change the opacity of the button.

- **ToolBarItems** - It is used to change the items that should be displayed in the zoom toolbar. By default, zoom-in, zoom-out, and reset buttons will be available. Other options include selection zoom and panning.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Load("load").ZoomSettings(zoom =>
zoom.Enable(true)).Layers(l =>
{
    l.ShapeSettings(sh=>sh.Fill("#C1DFF5")).ShapeData(ViewBag.worldMap).Add();
}).Render()
<script>
    function load(args) {
        var maps = document.getElementById('maps').ej2_instances[0];
        maps.zoomSettings.toolbarSettings.buttonSettings.fill = "pink";
        maps.zoomSettings.toolbarSettings.buttonSettings.padding = 10;
        maps.zoomSettings.toolbarSettings.buttonSettings.color = "red";
        maps.zoomSettings.toolbarSettings.buttonSettings.borderColor =
"green";
        maps.zoomSettings.toolbarSettings.buttonSettings.radius = 35;
        maps.zoomSettings.toolbarSettings.buttonSettings.selectionColor =
"#d55e5e";
        maps.zoomSettings.toolbarSettings.buttonSettings.selectionColor =
"#5ed59a";
        maps.zoomSettings.toolbarSettings.buttonSettings.opacity = 0.6;
        maps.zoomSettings.toolbarSettings.buttonSettings.borderWidth = 2;
        maps.zoomSettings.toolbarSettings.buttonSettings.toolbarItems =
['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'];
    }
</script>
```

ZOOMTOOLBARBUTTON.CS

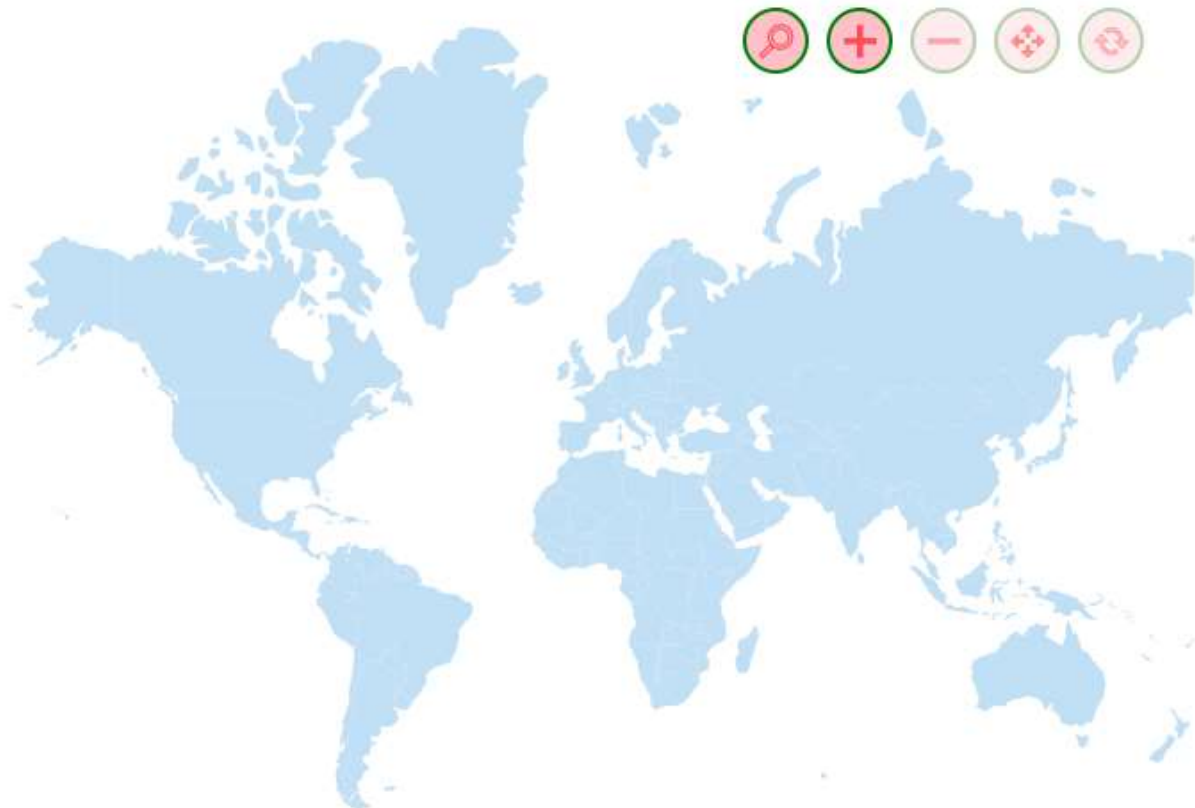
```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {

```

```

        string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



<!-- markdownlint-disable MD036 -->

Customizing the tooltip of the zoom toolbar

The appearance of the tooltip of the zoom toolbar can be customized by using the `TooltipSettings` option in the `ToolbarSettings` of the [ZoomSettings](#) property. The following properties are available to customize the zoom toolbar tooltip.

- **Visible** - Enables or disables the tooltip of the zoom toolbar.
- **Fill** - It is used to change the background color of the tooltip of the zoom toolbar.
- **BorderOpacity** - It is used to change the opacity of the border of the zoom toolbar's tooltip.
- **BorderWidth** - It is used to change the thickness of the border of the zoom toolbar's tooltip.

- **BorderColor** - It is used to change the color of the border of the zoom toolbar's tooltip.
- **FontColor** - It is used to change the color of the text in the tooltip of the zoom toolbar.
- **FontFamily** - It is used to change the font family of the text in the tooltip of the zoom toolbar.
- **FontStyle** - It is used to change the font style of the text in the tooltip of the zoom toolbar.
- **FontWeight** - It is used to change the font weight of the text in the tooltip of the zoom toolbar.
- **FontSize** - It is used to change the size of the text in the tooltip of the zoom toolbar.
- **FontOpacity** - It is used to change the opacity of the text in the tooltip of the zoom toolbar.

CSSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Load("load").ZoomSettings(zoom =>
zoom.Enable(true)).Layers(l =>
{
    1.ShapeSettings(sh=>sh.Fill("#C1DFF5")).ShapeData(ViewBag.worldMap).Add();
}).Render()
<script>
    function load(args) {
        var maps = document.getElementById('maps').ej2_instances[0];
        maps.zoomSettings.toolbarSettings.tooltipSettings.visible = true;
        maps.zoomSettings.toolbarSettings.tooltipSettings.borderWidth = 2;
        maps.zoomSettings.toolbarSettings.tooltipSettings.borderColor =
"green";
        maps.zoomSettings.toolbarSettings.tooltipSettings.fontColor =
"black";
        maps.zoomSettings.toolbarSettings.tooltipSettings.fill = "violet";
        maps.zoomSettings.toolbarSettings.tooltipSettings.fontFamily =
"Times New Roman";
        maps.zoomSettings.toolbarSettings.tooltipSettings.fontWeight = 200;
        maps.zoomSettings.toolbarSettings.tooltipSettings.fontSize = "22px";
        maps.zoomSettings.toolbarSettings.tooltipSettings.fontOpacity = 1;
        maps.zoomSettings.toolbarSettings.buttonSettings.toolbarItems =
['Zoom', 'ZoomIn', 'ZoomOut', 'Pan', 'Reset'];
    }
</script>
```

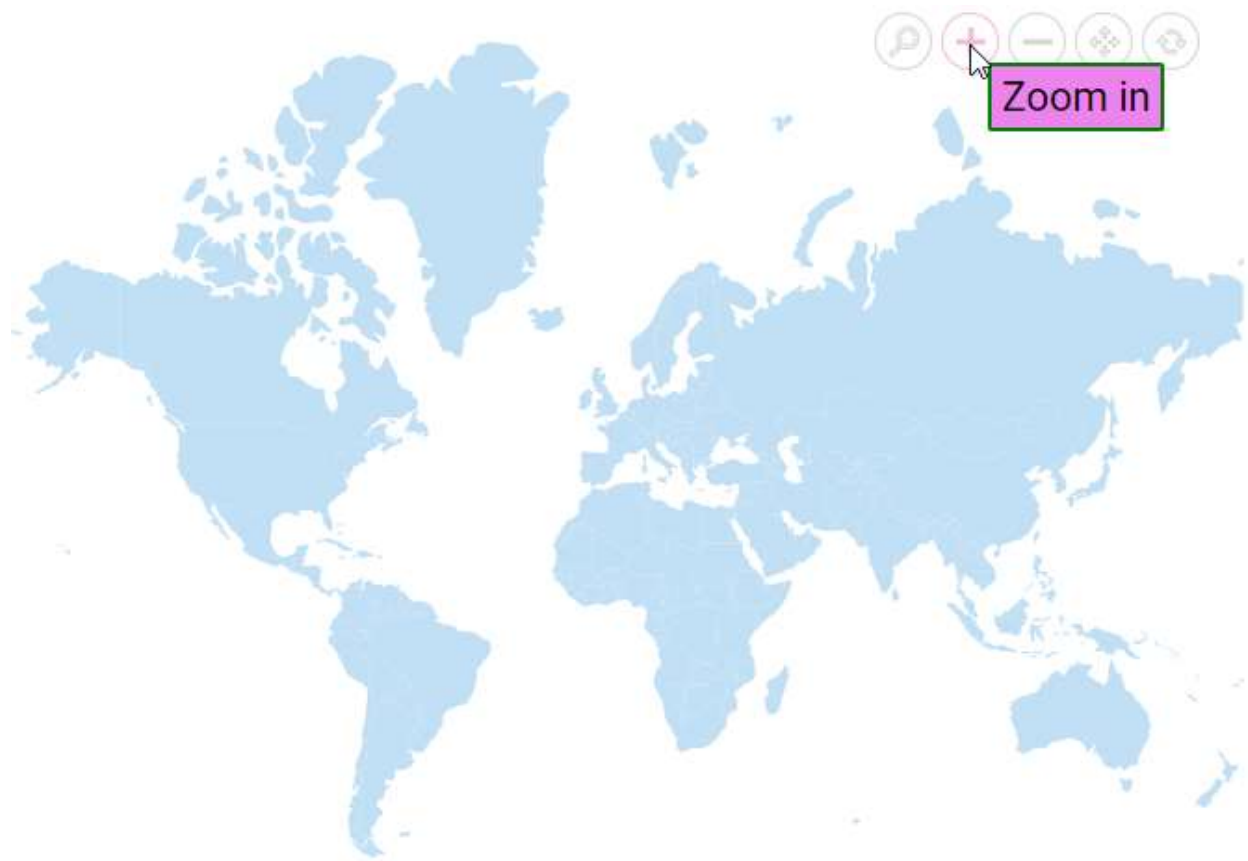
TOOLBARBUTTONTOOLTIP.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
        }
    }
}
```

```

        ViewBag.worldMap = GetMap();
        return View();
    }
    public object GetWorldMap()
    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
        return JsonConvert.DeserializeObject(allText);
    }
    public object GetMap()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



Selection

Each shape in the Maps can be selected and deselected during interaction with the shapes. Selection is enabled by setting the `Enable` property of `MapsSelectionSettings` to `true`.

The following properties are available to customize the selection of Maps elements such as shapes, bubbles, markers and legends.

- **Border** - To customize the color, width and opacity of the border of which element is selected in Maps.
- **Fill** - Applies the color for the element that is selected.
- **Opacity** - To customize the transparency for the element that is selected.
- **EnableMultiSelect** - To enable or disable the selection for multiple shapes or markers or bubbles in the Maps.

By tapping on the specific legend, the shapes which are bounded to the selected legend is also selected and vice versa.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var select = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    };
    var data = new[]
    {
        new { Country= "China", Membership= "Permanent" },
        new { Country= "France", Membership= "Permanent" },
        new { Country= "Russia", Membership= "Permanent" },
        new { Country= "Kazakhstan", Membership= "Non-Permanent" },
        new { Country= "Poland", Membership= "Non-Permanent" },
        new { Country= "Sweden", Membership= "Non-Permanent" }
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#D84444",Value= "Permanent" },
        new MapsColorMapping { Color= "#316DB5", Value = "Non-Permanent" }
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true)).Layers(l =>
{
    l.SelectionSettings(select)
    .ShapeSettings(s =>
s.ColorValuePath("Membership").ColorMapping(colormapping)).DataSource(data).
ShapeDataPath("Country").ShapePropertyPath("name").ShapeData(ViewBag.worldMa
p).Add();
}).Render()
```

SELECTION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Enable selection for bubbles

To enable the selection for bubbles in Maps, set the `MapsSelectionSettings` in `MapsBubble` and set the `Enable` property of `MapsSelectionSettings` as `true`.

CSHTML

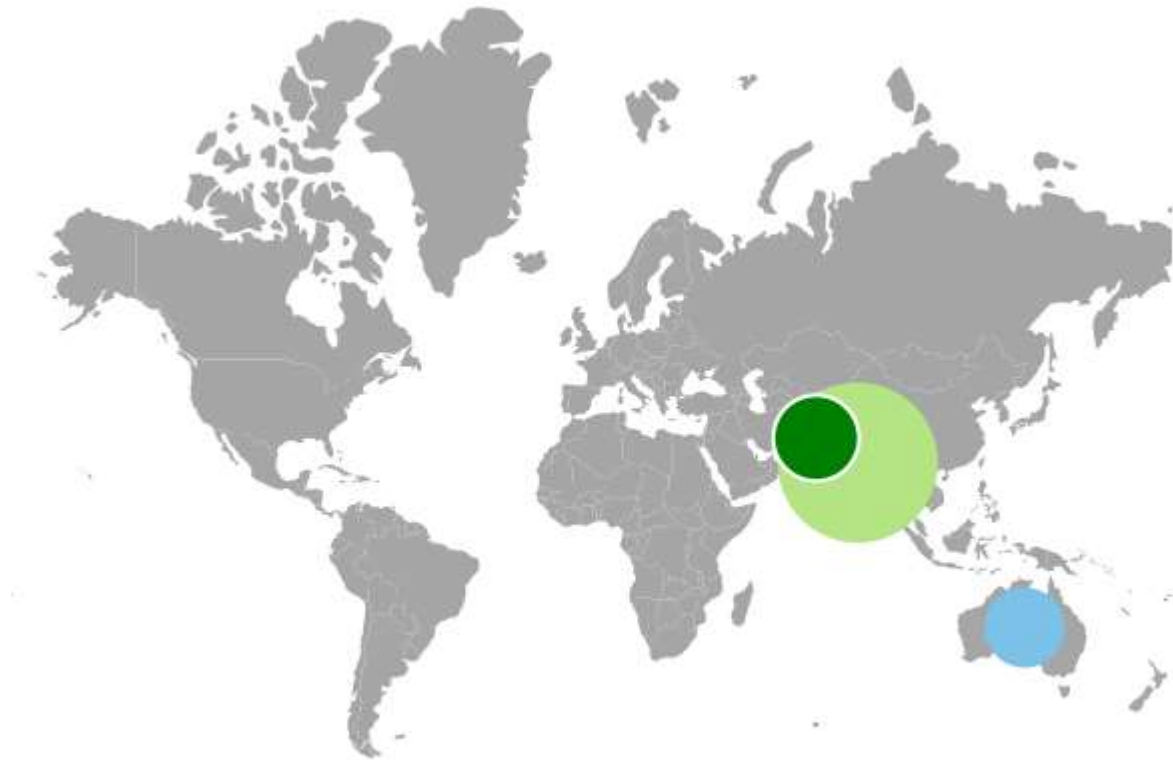
```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{ var select = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
            {
                Color = "white",
                Width = 2,
                Opacity = 1
            }
    };
}
@Html.EJS().Maps("maps").Layers(1 =>
```

```
{
    1
    .BubbleSettings(bubble =>
bubble.Visible(true).SelectionSettings(select).MinRadius(20).MaxRadius(20).Data
ataSource(ViewBag.bubbleData).ValuePath("population").Add()).

ShapeDataPath("name").ShapePropertyPath("name").ShapeData(ViewBag.worldMap).
Add();
}).Render()
```

SELECTION-BUBBLE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.worldMap = GetMap();
            List<BubbleData> data = new List<BubbleData>
            {
                new BubbleData { name= "India", population= "38332521" },
                new BubbleData { name= "Australia", population= "383521" },
                new BubbleData { name= "Pakistan", population= "3090416" }
            };
            ViewBag.bubbleData = data;
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
        public class BubbleData
        {
            public string name { get; set; }
            public string population { get; set; }
        }
    }
}
```

Enable selection for markers

To enable the selection for markers in Maps, set the `MapsSelectionSettings` in the `MapsMarker` and set the `Enable` property of the `MapsSelectionSettings` as `true`.

C#HTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var select = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    };
};
```

```

var data = new[]
{
    new {latitude= 37.0000, longitude= -120.0000, city= "California" },
    new {latitude= 40.7127, longitude= -74.0059, city= "New York" },
    new {latitude= 42.0000, longitude= -93.0000, city= "Iowa" }
};
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l
    .MarkerSettings(marker =>
marker.Visible(true).SelectionSettings(select).Shape(Syncfusion.EJ2.Maps.Mar
kerType.Balloon).
    DataSource(data).Height(20).Width(20).Add()
    .ShapeData(ViewBag.worldMap).Add();
}).Render()

```

SELECTION-MARKER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Enable selection for polygons

When the [Enable](#) property of [SelectionSettings](#) is set to **true**, the polygon shapes can be selected via user interaction. The following properties are available in [SelectionSettings](#) to customize the polygon shape when it is selected.

- [EnableMultiSelect](#) - It is used to enable multiple selection of polygon shapes.
- [Fill](#) - It is used to change the color of the selected polygon shape.
- [Opacity](#) - It is used to change the opacity of the selected polygon shape.
- [Border](#) - This property is used to change the color, width, and opacity of the border of the selected polygon shape.

The following example shows how to select the polygon shape in the geometry map.

C#HTML

```
@using Syncfusion.EJ2.Maps;  
@{  
    var data = new[]  
    {
```

```

    new { longitude = -1.8920678947185365, latitude = 35.06195799239681
  },
    new { longitude = -1.6479633699113947, latitude = 33.58989612266137
  },
    new { longitude = -1.4201220366858252, latitude = 32.819439646045254
  },
    new { longitude = -1.197974596225663, latitude = 32.26940895444655
  },
    new { longitude = -2.891112397949655, latitude = 32.10303058820031
  },
    new { longitude = -3.8246984550501963, latitude = 31.34551662687602
  },
    new { longitude = -3.720166273688733, latitude = 30.758086682848685
  },
    new { longitude = -5.6571886081189575, latitude = 29.613582597203006
  },
    new { longitude = -7.423353242214745, latitude = 29.44328441403087
  },
    new { longitude = -8.6048931685323, latitude = 28.761444633616776 },
    new { longitude = -8.695726975465703, latitude = 27.353491085576195
  },
    new { longitude = 3.837867279970908, latitude = 19.15916564839422 },
    new { longitude = 6.0705408799045415, latitude = 19.48749097192868
  },
    new { longitude = 12.055736352807713, latitude = 23.694596786078293
  },
    new { longitude = 11.272522332402986, latitude = 24.289329186946034
  },
    new { longitude = 10.30872578261932, latitude = 24.65419958524693 },
    new { longitude = 9.910236690050027, latitude = 25.48943950947175 },
    new { longitude = 9.432639882414293, latitude = 26.398372489836902
  },
    new { longitude = 9.898266456582292, latitude = 26.73489453809293 },
    new { longitude = 9.560243026853641, latitude = 30.31040379467153 },
    new { longitude = 8.943853847283322, latitude = 32.350324876652195
  },
    new { longitude = 7.57004059025715, latitude = 33.75071049019398 },
    new { longitude = 8.0906322609153, latitude = 34.69043151009983 },
    new { longitude = 8.363285449347273, latitude = 35.38654406371319 },
    new { longitude = 8.26139549449448, latitude = 36.44751078733985 },
    new { longitude = 8.61100824823302, latitude = 36.881913362940196 },
    new { longitude = 7.4216488925819135, latitude = 37.021408008916254
  },
    new { longitude = 6.461182254165351, latitude = 36.99092409199429 },
    new { longitude = 5.297178918070159, latitude = 36.69985479014656 },
    new { longitude = 3.6718056161224695, latitude = 36.86470546831693
  },
    new { longitude = 1.2050052555659931, latitude = 36.57658056301722
  },
    new { longitude = -0.26968570003779746, latitude =
35.806903541813625 },
    new { longitude = -0.995191786435754, latitude = 35.58466127904214
  },
    new { longitude = -1.8920678947185365, latitude = 35.06195799239681
  }
};
var polygons = new List<Syncfusion.EJ2.Maps.MapsPolygon>

```

```

    {
        new Syncfusion.EJ2.Maps.MapsPolygon{ Points=data, Fill="blue",
        Opacity=0.7, BorderColor="green", BorderOpacity=0.7, BorderWidth=2 }
    };
    var highlight = new Syncfusion.EJ2.Maps.MapsHighlightSettings
    {
        Enable = true, Fill = "blue", Opacity = 0.7,
        Border = new MapsBorder
        {
            Color ="green",
            Opacity = 0.7,
            Width=2
        }
    };
    var selection = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "violet",
        EnableMultiSelect = false,
        Opacity = 0.8,
        Border = new MapsBorder
        {
            Color = "cyan",
            Opacity = 1,
            Width = 7
        }
    };
}
@(Html.EJS().Maps("maps").Layers(layers => { layers.PolygonSettings(polygon
=> {
    polygon.Polygons(polygons).HighlightSettings(highlight).SelectionSettings(se
lection); }).ShapeData(ViewBag.world_map).Add(); }).Render())

```

POLYGON-SHAPE-SELECTION.CS

```

using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        // To access the data in Core
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        // To access the data in MVC
        public object GetMap()

```

```

    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



Public method for the shape selection

The `shapeSelection` method can be used to select each shape in the Maps. `LayerIndex`, `propertyName`, country name, and selected value as a boolean state(true / false) are the input parameters for this method.

CSHTML

```

@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var select = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    }
}

```

```

    };
}
<button id="selection">Selection</button>
<button id="unselection">UnSelection</button>
@Html.EJS().Maps("maps").Load("onMapLoad").Layers(l =>
{
    l.SelectionSettings(select).ShapeData(ViewBag.worldMap).Add();
}).Render()
<script>
    function onMapLoad(args) {
        window.maps = args.maps;
    }
    window.onload = function () {
        document.getElementById('selection').onclick = () => {
            window.maps.shapeSelection(0, "continent", "Asia", true);
        };
        document.getElementById('unselection').onclick = () => {
            window.maps.shapeSelection(0, "continent", "Asia", false);
        };
    };
};
</script>

```

SELECTION-METHOD.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Initial shape selection

The shape is initially selected using the `MapsInitialShapeSelection`, and the values are mapped to the `ShapePath` and `ShapeValue`.

CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.worldMap,
        SelectionSettings = new MapsSelectionSettings
        {
            Enable= true,
            Fill= "green",
            Border = new MapsBorder
            {
                Color = "white",
                Width = 2,
```



```

                Opacity = 1
            }
        },
        InitialShapeSelection = new
List<Syncfusion.EJ2.Maps.MapsInitialShapeSelection>
    {
        new Syncfusion.EJ2.Maps.MapsInitialShapeSelection{
ShapePath= "continent", ShapeValue= "Africa" },
        new Syncfusion.EJ2.Maps.MapsInitialShapeSelection{
ShapePath= "name", ShapeValue= "India" }
    }
    }).Render()

```

INITIAL-SHAPE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using EJ2MVCSampleBrowser.Models;
namespace EJ2MVCSampleBrowser.Controllers.Maps
{
    public partial class MapsController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Initial marker selection

Using the `InitialMarkerSelection`, the marker shape can be selected initially. Markers render based on the `Latitude` and `Longitude` values.

C#HTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var select = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    };
    var point = new double[] { 42.0000, -93.0000 };
    var data = new[]
```

```

        {
            new {latitude= 37.0000, longitude= -120.0000, city= "California" },
            new {latitude= 40.7127, longitude= -74.0059, city= "New York" },
            new {latitude= 42.0000, longitude= -93.0000, city= "Iowa" }
        };
    }
    @Html.EJS().Maps("maps").Layers(l =>
    {
        l
        .MarkerSettings(marker =>
        marker.Visible(true).SelectionSettings(select).InitialMarkerSelection(point)
        .
        Shape(Syncfusion.EJ2.Maps.MarkerType.Balloon).
        DataSource(data).Height(20).Width(20).Fill("green").Add()
        .ShapeData(ViewBag.worldMap).Add();
    }).Render()
    )

```

INITIALMARKERSELECTION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```

Highlight

Each shape in the Maps can be highlighted during mouse hover on the Maps elements such as shapes, bubbles, markers and legends. Highlight is enabled by setting the **Enable** property of **MapsHighlightSettings** to **true**.

The following properties are available to customize the highlight of Maps elements such as shapes, bubbles, markers and legends.

- **Border** - To customize the color, width and opacity of the border of which element is highlighted in Maps.
- **Fill** - Applies the color for the element that is highlighted.
- **Opacity** - To customize the transparency for the element that is highlighted.

Hovering on the specific legend, the shapes which are bounded to the selected legend is also highlighted and vice versa.

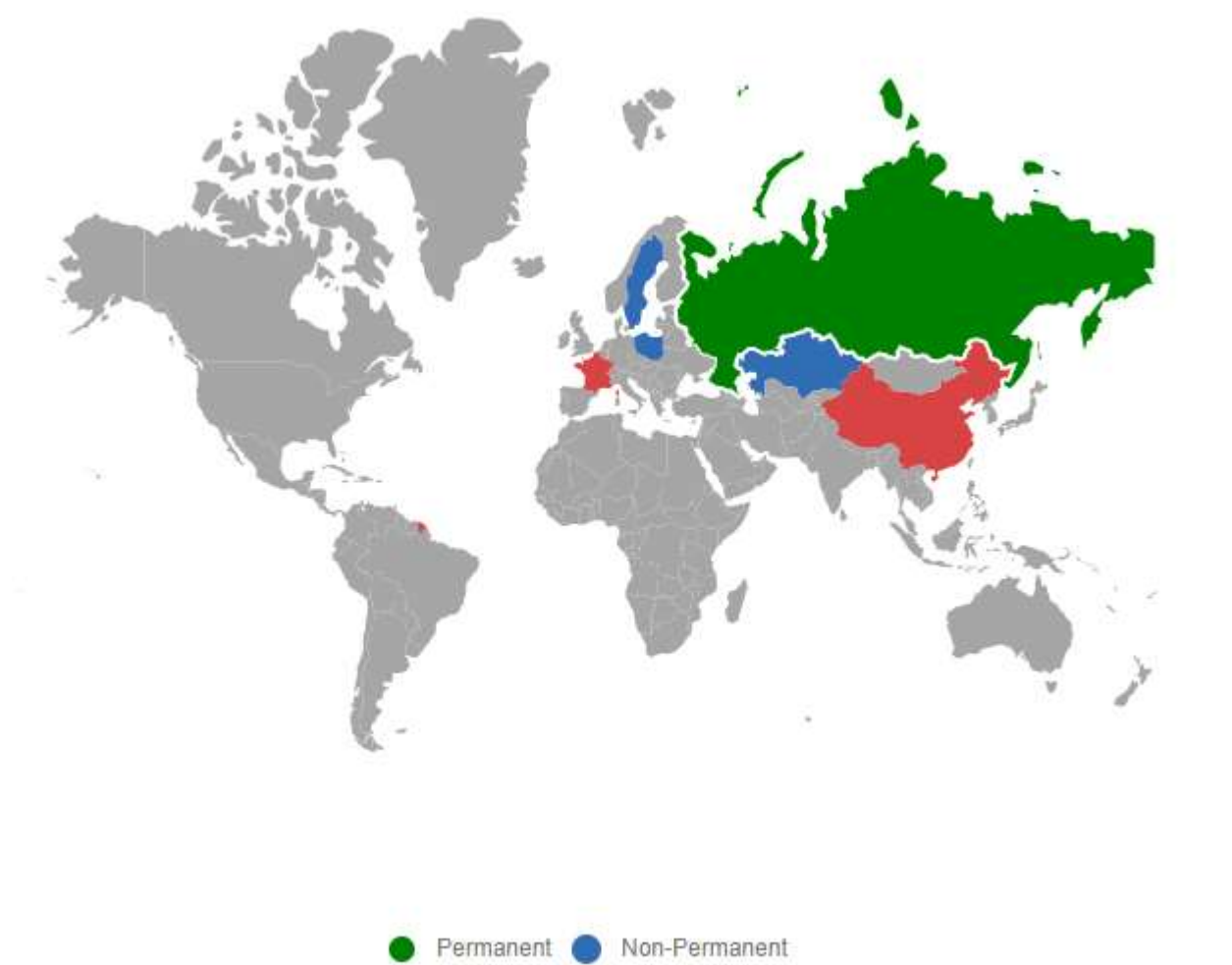
CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var highlight = new Syncfusion.EJ2.Maps.MapsHighlightSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    };
    var data = new[]
    {
        new { Country= "China", Membership= "Permanent" },
        new { Country= "France", Membership= "Permanent" },
        new { Country= "Russia", Membership= "Permanent" },
        new { Country= "Kazakhstan", Membership= "Non-Permanent" },
        new { Country= "Poland", Membership= "Non-Permanent" },
        new { Country= "Sweden", Membership= "Non-Permanent" }
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#D84444",Value= "Permanent" },
        new MapsColorMapping { Color= "#316DB5", Value = "Non-Permanent" }
    };
}
@Html.EJS().Maps("maps").LegendSettings(legend =>
legend.Visible(true)).Layers(l =>
{
    l.HighlightSettings(highlight)
    .ShapeSettings(s =>
s.ColorValuePath("Membership").ColorMapping(colormapping)).DataSource(data).
ShapeDataPath("Country").ShapePropertyPath("name").ShapeData(ViewBag.worldMap).Add();
})
```

```
}).Render()
```

HIGHLIGHT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/worldmap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Enable highlight for bubbles

To enable the highlight for bubbles in Maps, set the `MapsHighlightSettings` in `MapsBubble` and set the `Enable` property of `MapsHighlightSettings` as **true**.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var highlight = new Syncfusion.EJ2.Maps.MapsHighlightSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    };
    var data = new[]
```

```

        {
            new { name= "India", population= "38332521" },
            new { name= "Australia", population= "383521" },
            new { name= "Pakistan", population= "3090416" }
        };
    }
    @Html.EJS().Maps("maps").Layers(l =>
    {
        l
        .BubbleSettings(bubble =>
        bubble.Visible(true).MinRadius(20).MaxRadius(20).DataSource(data).ValuePath(
        "population").HighlightSettings(highlight).Add()).

        ShapeDataPath("name").ShapePropertyPath("name").ShapeData(ViewBag.worldMap).
        Add();
    }).Render()

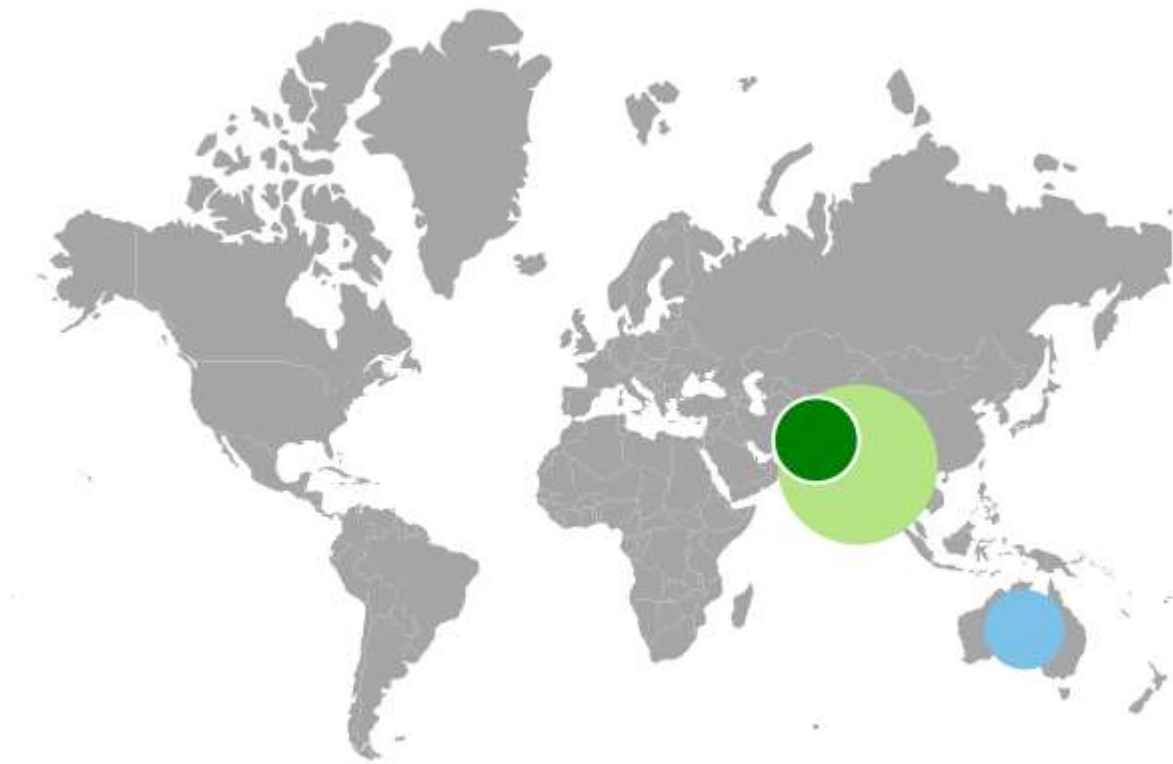
```

HIGHLIGHT-BUBBLE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/worldmap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
            System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Enable highlight for markers

To enable the highlight for markers in Maps, set the `MapsHighlightSettings` in `MapsMarker` and set the `Enable` property of `MapsHighlightSettings` as `true`.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var highlight = new Syncfusion.EJ2.Maps.MapsHighlightSettings
    {
        Enable = true,
        Fill = "green",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "white",
            Width = 2,
            Opacity = 1
        }
    };
    var data = new[]
    {
```



```

        new {latitude= 37.0000, longitude= -120.0000, city= "California" },
        new {latitude= 40.7127, longitude= -74.0059, city= "New York" },
        new {latitude= 42.0000, longitude= -93.0000, city= "Iowa" }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l
    .MarkerSettings(marker =>
marker.Visible(true).HighlightSettings(highlight).Shape(Syncfusion.EJ2.Maps.
MarkerType.Balloon).
        DataSource(data).Height(20).Width(20).Add())
    .ShapeData(ViewBag.worldMap).Add();
}).Render()

```

HIGHLIGHT-MARKER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/worldmap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



Enable highlight for polygons

The polygon shapes can be highlighted via user interaction if the [Enable](#) property of [HighlightSettings](#) is set to **true**. The following properties are available in [HighlightSettings](#) to customize the polygon shape when it is highlighted.

- [Fill](#) - It is used to change the color of the highlighted polygon shape.
- [Opacity](#) - It is used to change the opacity of the highlighted polygon shape.
- [Border](#) - This property is used to change the color, width, and opacity of the border of the highlighted polygon shape.

The following example shows how to highlight a polygon shape on a geometry map.

C#HTML

```
@using Syncfusion.EJ2.Maps;
@{
    var data = new[]
    {
        new { longitude = -1.8920678947185365, latitude = 35.06195799239681
    },
```

```

    new { longitude = -1.6479633699113947, latitude = 33.58989612266137
  },
    new { longitude = -1.4201220366858252, latitude = 32.819439646045254
  },
    new { longitude = -1.197974596225663, latitude = 32.26940895444655
  },
    new { longitude = -2.891112397949655, latitude = 32.10303058820031
  },
    new { longitude = -3.8246984550501963, latitude = 31.34551662687602
  },
    new { longitude = -3.720166273688733, latitude = 30.758086682848685
  },
    new { longitude = -5.6571886081189575, latitude = 29.613582597203006
  },
    new { longitude = -7.423353242214745, latitude = 29.44328441403087
  },
    new { longitude = -8.6048931685323, latitude = 28.761444633616776 },
    new { longitude = -8.695726975465703, latitude = 27.353491085576195
  },
    new { longitude = 3.837867279970908, latitude = 19.15916564839422 },
    new { longitude = 6.0705408799045415, latitude = 19.48749097192868
  },
    new { longitude = 12.055736352807713, latitude = 23.694596786078293
  },
    new { longitude = 11.272522332402986, latitude = 24.289329186946034
  },
    new { longitude = 10.30872578261932, latitude = 24.65419958524693 },
    new { longitude = 9.910236690050027, latitude = 25.48943950947175 },
    new { longitude = 9.432639882414293, latitude = 26.398372489836902
  },
    new { longitude = 9.898266456582292, latitude = 26.73489453809293 },
    new { longitude = 9.560243026853641, latitude = 30.31040379467153 },
    new { longitude = 8.943853847283322, latitude = 32.350324876652195
  },
    new { longitude = 7.57004059025715, latitude = 33.75071049019398 },
    new { longitude = 8.0906322609153, latitude = 34.69043151009983 },
    new { longitude = 8.363285449347273, latitude = 35.38654406371319 },
    new { longitude = 8.26139549449448, latitude = 36.44751078733985 },
    new { longitude = 8.61100824823302, latitude = 36.881913362940196 },
    new { longitude = 7.4216488925819135, latitude = 37.021408008916254
  },
    new { longitude = 6.461182254165351, latitude = 36.99092409199429 },
    new { longitude = 5.297178918070159, latitude = 36.69985479014656 },
    new { longitude = 3.6718056161224695, latitude = 36.86470546831693
  },
    new { longitude = 1.2050052555659931, latitude = 36.57658056301722
  },
    new { longitude = -0.26968570003779746, latitude =
35.806903541813625 },
    new { longitude = -0.995191786435754, latitude = 35.58466127904214
  },
    new { longitude = -1.8920678947185365, latitude = 35.06195799239681
  }
  };
  var polygons = new List<Syncfusion.EJ2.Maps.MapsPolygon>
  {

```

```

        new Syncfusion.EJ2.Maps.MapsPolygon{ Points=data, Fill="blue",
        Opacity=0.7, BorderColor="green", BorderOpacity=0.7, BorderWidth=2 }
    };
    var highlight = new Syncfusion.EJ2.Maps.MapsHighlightSettings
    {
        Enable = true, Fill = "yellow", Opacity = 0.4,
        Border = new MapsBorder
        {
            Color = "blue",
            Opacity = 0.6,
            Width=4
        }
    };

    var selection = new Syncfusion.EJ2.Maps.MapsSelectionSettings
    {
        Enable = true,
        Fill = "red",
        EnableMultiSelect = false,
        Opacity = 0.7,
        Border = new MapsBorder
        {
            Color = "green",
            Opacity = 0.7,
            Width = 2
        }
    };
}
@(Html.EJS().Maps("maps").Layers(layers => { layers.PolygonSettings(polygon
=> {
    polygon.Polygons(polygons).HighlightSettings(highlight).SelectionSettings(se
lection); }).ShapeData(ViewBag.world_map).Add(); }).Render())

```

POLYGON-SHAPE-HIGHLIGHT.CS

```

using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.world_map = GetMap();
            return View();
        }
        // To access the data in Core
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        // To access the data in MVC
        public object GetMap()

```

```

    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
}

```



Tooltip

On mouse over or touch end event, the tooltip is used to get more information about the layer, bubble, or marker. It can be enabled separately for layer or bubble or marker by using the **Visible** property of **MapsTooltipSettings** as **true**. The **ValuePath** property in the tooltip takes the field name that presents in data source and displays that value as tooltip text. The **TooltipDisplayMode** property is used to change the display mode of the tooltip in Maps. Following display modes of tooltip are available in the Maps component. By default, **TooltipDisplayMode** is set to **MouseMove**.

CSHTML

```

@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Layers(l =>
{
    l.TooltipSettings(ts =>ts.ValuePath("name").Visible(true))
    .ShapeData(ViewBag.worldMap).Add();
}).Render()

```

TOOLTIP.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Customization

The following properties are available in the `MapsTooltipSettings` to customize the tooltip of the Maps component.

- **Border** - To customize the color, width and opacity of the border of the tooltip in layers, markers, and bubbles of Maps.
- **Fill** - Applies the color of the tooltip in layers, markers, and bubbles of Maps.
- **Format** - To customize the format of the tooltip in layers, markers, and bubbles of Maps
- **TextStyle** - To customize the style of the text in the tooltip for layers, markers, and bubbles of Maps.

CSHTML

```
@using Syncfusion.EJ2;  
@using Syncfusion.EJ2.Maps;  
@{  
    var tooltip = new Syncfusion.EJ2.Maps.MapsTooltipSettings  
    {  
        Visible = true,  
        ValuePath = "name",  
        Format = "${name}: ${value1}",  
        Fill = "#D0D0D0",  
        TextStyle = new MapsFont
```

```

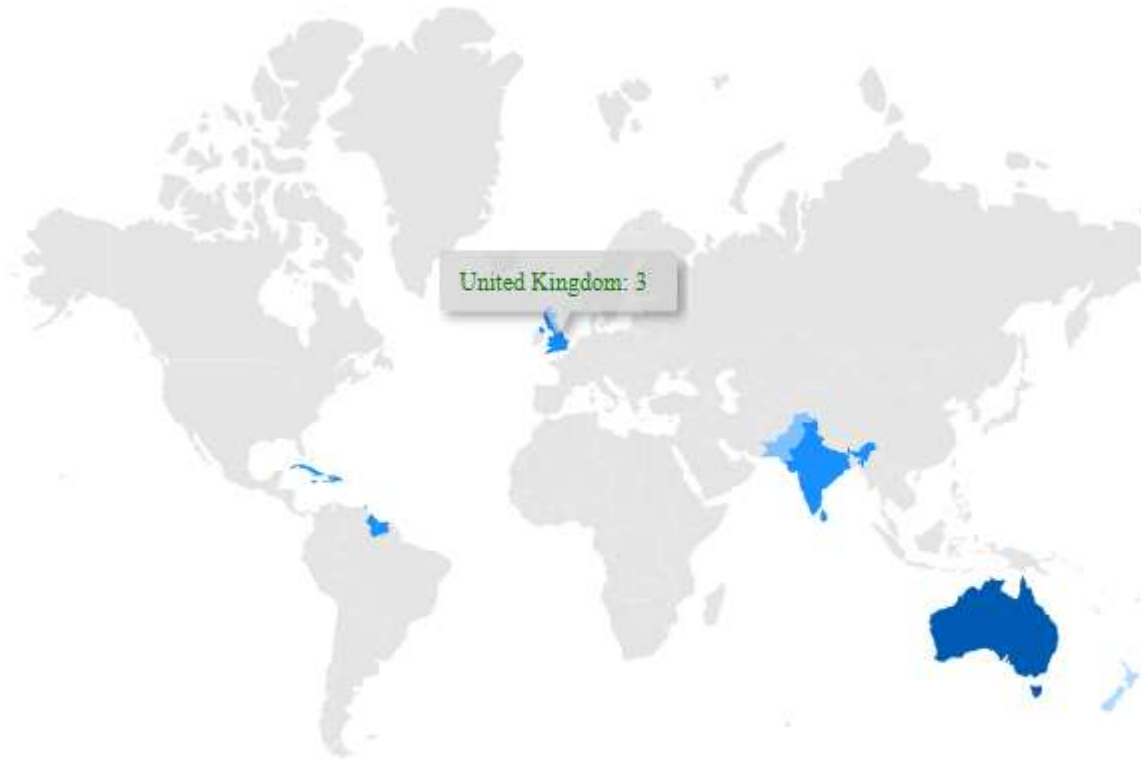
        {
            Color = "green",
            FontFamily = "Times New Roman",
            FontStyle = "Sans-serif"
        }
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#b3daff",Value= "1" },
        new MapsColorMapping { Color= "#80c1ff", Value = "2" },
        new MapsColorMapping { Color= "#1a90ff", Value = "3" },
        new MapsColorMapping { Color= "#005cb3", Value = "7" }
    };
    var data = new[]
    {
        new { name= "India", value1= "3", value2="2", country="India" },
        new { name= "Dominican Rep.", value1= "3", value2="2", country="West
Indies" },
        new { name= "Cuba", value1= "3", value2="2", country="West Indies"
    },
        new { name= "Jamaica", value1= "3", value2="2", country="West
Indies" },
        new { name= "Haiti", value1= "3", value2="2", country="West Indies"
    },
        new { name= "Guyana", value1= "3", value2="2", country="West Indies"
    },
        new { name= "Suriname", value1= "3", value2="2", country="West
Indies" },
        new { name= "Trinidad and Tobago", value1= "3", value2="2",
country="West Indies" },
        new { name= "Sri Lanka", value1= "3", value2="1", country="Sri
Lanka" },
        new { name= "United Kingdom", value1= "3", value2="0",
country="England" },
        new { name= "Pakistan", value1= "2", value2="1", country="Pakistan"
    },
        new { name= "New Zealand", value1= "1", value2="0", country="New
Zealand" },
        new { name= "Australia", value1= "7", value2="5",
country="Australia" },
    };
}
@Html.EJS().Maps("maps").TooltipRender("tooltipRender").Layers(l =>
{
    l.ShapeSettings(shape =>
shape.Fill("#E5E5E5").ColorValuePath("value1").ColorMapping(colormapping))

    .ShapeDataPath("name").ShapePropertyPath("name").DataSource(data).TooltipSet
tings(tooltip).ShapeData(ViewBag.worldMap).Add();
}).Render()
<script>
    function tooltipRender(args) {
        if (!args.options['data']) {
            args.cancel = true;
        }
    }
</script>

```


TOOLTIP-CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Tooltip template

The HTML element can be rendered in the tooltip of the Maps using the **Template** property of the **MapsTooltipSettings**. In the following example, `#{value1}` and `#{value2}` are the place holders in the HTML element to display the value1 and value2 values of the corresponding shape.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var tooltip = new Syncfusion.EJ2.Maps.MapsTooltipSettings
    {
        Visible = true,
        ValuePath = "name",
        Template = "#template"
    };
    var colormapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping{ Color = "#b3daff",Value= "1" },
        new MapsColorMapping { Color= "#80c1ff", Value = "2" },
        new MapsColorMapping { Color= "#1a90ff", Value = "3" },
        new MapsColorMapping { Color= "#005cb3", Value = "7" }
```

```

};
var data = new[]
{
    new { name= "India", value1= "3", value2="2", country="India" },
    new { name= "Dominican Rep.", value1= "3", value2="2", country="West
Indies" },
    new { name= "Cuba", value1= "3", value2="2", country="West Indies"
},
    new { name= "Jamaica", value1= "3", value2="2", country="West
Indies" },
    new { name= "Haiti", value1= "3", value2="2", country="West Indies"
},
    new { name= "Guyana", value1= "3", value2="2", country="West Indies"
},
    new { name= "Suriname", value1= "3", value2="2", country="West
Indies" },
    new { name= "Trinidad and Tobago", value1= "3", value2="2",
country="West Indies" },
    new { name= "Sri Lanka", value1= "3", value2="1", country="Sri
Lanka" },
    new { name= "United Kingdom", value1= "3", value2="0",
country="England" },
    new { name= "Pakistan", value1= "2", value2="1", country="Pakistan"
},
    new { name= "New Zealand", value1= "1", value2="0", country="New
Zealand" },
    new { name= "Australia", value1= "7", value2="5",
country="Australia" },
};
}
@Html.EJS().Maps("maps").TooltipRender("tooltipRender").Layers(l =>
{
    l.ShapeSettings(shape =>
shape.Fill("#E5E5E5").ColorValuePath("value1").ColorMapping(colormapping))

    .ShapeDataPath("name").ShapePropertyPath("name").DataSource(data).TooltipSet
tings(tooltip).ShapeData(ViewBag.worldMap).Add();
}).Render()
<div id="template" style="display:none">
    <div class="toolback">
        <div class="listing2">
            <center>
                ${country}
            </center>
        </div>
        <hr style="margin-top: 2px;margin-bottom:5px;border:0.5px solid
#DDDDDD">
        <div>
            <span class="listing1">Finalist : </span><span
class="listing2">${value1}</span>
        </div>
        <div>
            <span class="listing1">Win : </span><span
class="listing2">${value2}</span>
        </div>
    </div>
</div>

```

```

</div>
<script>
    function tooltipRender(args) {
        if (!args.options['data']) {
            args.cancel = true;
        }
    }
</script>
<style>
    .toolback {
        border - radius: 4px;
        border: 1px #abb9c6;
        opacity: 90%;
        background: rgba(53, 63, 76, 0.90);
        box-shadow: 0px 2px 2px rgba(0, 0, 0, 0.40);
        padding-bottom: 5px;
        padding-top: 10px;
        padding-left: 10px;
        padding-right: 10px;
        width: 90px;
    }
    .listing1 {
        font - size:13px;
        color: #cccccc
    }
    .listing2 {
        font - size:13px;
        color: #ffffff;
        font-weight: 500;
    }
</style>

```

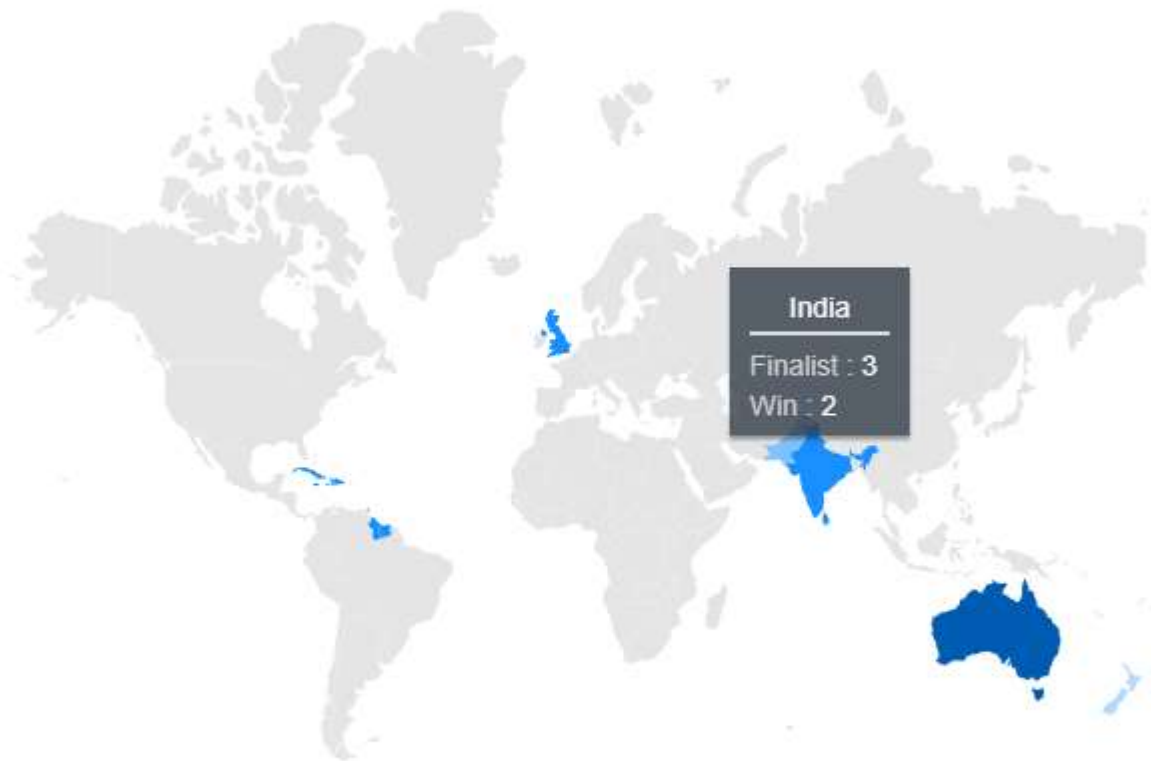
TOOLTIP-TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");

```

```
        return JsonConvert.DeserializeObject(allText);  
    }  
    public object GetMap()  
    {  
        string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
        return JsonConvert.DeserializeObject(allText, typeof(object));  
    }  
}
```



Print and Export in ASP.NET MVC Maps component

Print

The rendered maps can be printed directly from the browser by calling the [print](#) method. To use the print functionality, set the [AllowPrint](#) property to **true**.

CSHTML

```
@using Syncfusion.EJ2;
```

```
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Print").CssClass("e-flat").Render()
@Html.EJS().Maps("maps").Height("450px").AllowPrint(true).Width("650px").Layers(l =>
{
    l.TooltipSettings(ts => ts.ValuePath("name").Visible(true))
    .DataLabelSettings(ds =>
ds.Visible(true).LabelPath("name").SmartLabelMode(Syncfusion.EJ2.Maps.SmartLabelMode.Trim))
    .ShapeSettings(ss =>
ss.Autofill(true)).ShapeData(ViewBag.ShapeData).Add();
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var map = document.getElementById('maps').ej2_instances[0];
        map.print();
    };
</script>
```

PRINT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.ShapeData = this.GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```

Export

Image Export

To use the image export functionality, we should set the [AllowImageExport](#) property to **true**. The rendered maps can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The maps can be exported as an image in the following formats.

- JPEG

- PNG
- SVG

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().Maps("maps").Height("450px").AllowImageExport(true).Width("650px")
.Layers(l => {
    l.TooltipSettings(ts => ts.ValuePath("name").Visible(true))
    .DataLabelSettings(ds =>
ds.Visible(true).LabelPath("name").SmartLabelMode(Syncfusion.EJ2.Maps.SmartL
abelMode.Trim))
    .ShapeSettings(ss
=>ss.Autofill(true)).ShapeData(ViewBag.ShapeData).Add();
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var map = document.getElementById('maps').ej2_instances[0];
        map.export('PNG', 'Export');
    };
</script>
```

EXPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.ShapeData = this.GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```

We can get the image file as base64 string for the JPEG and PNG formats. The maps can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file

name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().Maps("maps").Height("450px").AllowImageExport(true).Width("650px")
.Layers(l => {
    l.TooltipSettings(ts => ts.ValuePath("name").Visible(true))
    .DataLabelSettings(ds =>
ds.Visible(true).LabelPath("name").SmartLabelMode(Syncfusion.EJ2.Maps.SmartL
abelMode.Trim))
    .ShapeSettings(ss
=>ss.Autofill(true)).ShapeData(ViewBag.ShapeData).Add();
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var map = document.getElementById('maps').ej2_instances[0];
        map.export('PNG', 'Export');
    };
</script>
```

EXPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.ShapeData = this.GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```

PDF Export

To use the PDF export functionality, we should set the [AllowPdfExport](#) property to **true**. The rendered maps can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters:

file type, file name and orientation of the PDF document. The orientation setting is optional and **0** indicates portrait and **1** indicates landscape.

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().Maps("maps").Height("450px").AllowImageExport(true).Width("650px")
.Layers(l => {
    l.TooltipSettings(ts => ts.ValuePath("name").Visible(true))
    .DataLabelSettings(ds =>
ds.Visible(true).LabelPath("name").SmartLabelMode(Syncfusion.EJ2.Maps.SmartL
abelMode.Trim))
    .ShapeSettings(ss
=>ss.Autofill(true)).ShapeData(ViewBag.ShapeData).Add();
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var map = document.getElementById('maps').ej2_instances[0];
        map.export('PNG', 'Export');
    };
</script>
```

EXPORT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.ShapeData = this.GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```

Export the tile maps

The rendered map with providers such as OSM, Bing and Google static maps can be exported using the [export](#) method. It supports the following export formats.

- JPEG
- PNG
- PDF

CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Export").CssClass("e-flat").Render()
@Html.EJS().Maps("maps").AllowImageExport(true).Layers(l=> {
    l.LayerType(Syncfusion.EJ2.Maps.ShapeLayerType.OSM).Add();
}).Render()
<script>
    document.getElementById('togglebtn').onclick = function () {
        var map = document.getElementById('maps').ej2_instances[0];
        map.export('PNG', 'Export');
    };
</script>
```

EXPORT-TILE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.shapeData = this.GetWorldMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
    }
}
```

State Persistence

State persistence

For state maintenance, state persistence allows Maps to save the current model value in browser cookies. This action is handled through the `EnablePersistence` property which is set to **false** by default. When this property is set to true, some of the Maps component model values are preserved even after the page is refreshed.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@Html.EJS().Maps("container").EnablePersistence(true).TitleSettings(title =>
    title.Text("Top 50 largest cities in the World").
        TextStyle(new MapsFont { Size = "16px" })).ZoomSettings(new
Syncfusion.EJ2.Maps.MapsZoomSettings
{
    Enable = true
}).Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeSettings = new MapsShapeSettings
        {
            Fill = "#C1DFF5"
        },
        ShapeData = ViewBag.shapeData,
        MarkerSettings = new
List<Syncfusion.EJ2.Maps.MapsMarker>
{
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = ViewBag.ClusterData,
                Shape=MarkerType.Circle,
                Height= 20,
                Width= 20,
                AnimationDuration= 0,
            }
        }
    }
}).Render()

```

PERSISTENCE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.Maps;
using System.Web.Script.Serialization;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using EJ2MVCSampleBrowser.Models;
namespace EJ2MVCSampleBrowser.Controllers.Maps
{
    public partial class MapsController : Controller
    {
        // GET: MarkerClustering
        public ActionResult MarkerClustering()
        {
            ViewBag.shapeData = this.getWorldMap();
            ViewBag.ClusterData = this.ClusterData();
            return View();
        }
    }
}

```

```

    }
    public object ClusterData()
    {
        string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/MapData/ClusterData.js
on"));
        return JsonConvert.DeserializeObject(allText, typeof(object));
    }
    public object getWorldMap()
    {
        string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
        return JsonConvert.DeserializeObject(allText);
    }
}
}

```

Accessibility in ASP.NET MVC Maps component

Maps has built-in accessibility features like screen reading, keyboard navigation, and WAI-ARIA attributes.

WAI-ARIA attributes

To meet accessibility standards, the Maps component follows to the [WAI-ARIA](#) patterns. In the Maps component, the following ARIA attributes are used:

| Attributes | Purpose |
|--------------------|--|
| --- | --- |
| role=region | It specifies the Maps areas that do not support interactive functions like selection and highlight. |
| role=button | It specifies the Maps areas where interactive functions such as selection and highlight are available. |
| aria-label | Provides an accessible name for Maps elements such as geometric map shapes, legend title, data labels, and so on. To learn more, see the next topic. |

Screen reading in Maps

Accessibility in the Maps component ensures that all users, regardless of ability or disability, can use screen reading. The following Map elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |
|---------------------|--|
| --- | --- |
| Shapes in the layer | Reads the names of the geographical shapes (such as countries, states, and regions) that appear on the Maps. |
| Legend title | Reads the contents of the legend's title as specified in Maps. |
| Legend item label | Reads the label of a legend item in Maps. |
| Data label | Reads the label specified for the shapes in the Maps layer. |
| Annotation | Reads the content specified in the annotation. |

- | Marker template | Reads the content provided in the marker template. |
- | Tooltip template | Reads the content provided in the tooltip template. |
- | Data label template | Reads the content provided in the data label template. |

Keyboard Navigation

All the Maps actions can be controlled via keyboard keys. The applicable key combinations and their relative functionalities are listed below for the appropriate UI features available in the component.

Interaction Keys | Description

Tab | Moves to the next focusable element on the map, such as the legend or shape.

Shift + Tab | Moves to the previous focusable element on the map, such as the legend or shape.

+ | When zooming is enabled, zoom in operation can be performed.

- | When zooming is enabled, zoom out operation can be performed.

Left arrow | When zoomed in, the map can be scrolled to the left.

Right arrow | When zoomed in, the map can be scrolled to the right.

Up arrow | When zoomed in, the map can be scrolled upward.

Down arrow | When zoomed in, the map can be scrolled downward.

R | When zooming is enabled, reset operation can be performed.

Enter | The page can be navigated to the next and previous states in legend. Similarly, the selection can be made while navigating over the shape.

Ensuring accessibility

The Maps component's accessibility levels are ensured using an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Maps component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Maps component with accessibility tools.

See also

- [Accessibility in Syncfusion Maps components](#)

Internationalization

Maps provide support for internationalization for the below elements.

- Data label
- Tooltip

For more information about number and date formatter, refer to the [internationalization](#) section.

<!-- markdownlint-disable MD036 -->

Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales. Internationalization library is used to globalize number, date, time values in Maps component using `Format` property in the `Maps`.

Numeric format

The numeric formats such as currency, percentage and so on can be displayed in the tooltip and data labels of the Maps using the `Format` property in the `Maps`. In the below example, the tooltip is globalized to **German** culture. When setting the `UseGroupingSeparator` property as **true**, the numeric text in the Maps separates with the comma separator.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var internData = new[]
    {
        new { Country= "China", Membership="Permanent", population=
"38332521" },
        new { Country= "France", Membership="Permanent", population=
"19651127" },
        new { Country= "Russia", Membership="Permanent", population=
"3090416" },
        new { Country= "Kazakhstan", Membership="Non-Permanent", population=
"1232521" },
        new { Country= "Poland", Membership="Non-Permanent", population=
"90332521" },
        new { Country= "Sweden", Membership="Non-Permanent", population=
"383521" }
    };
    var colorMapping = new List<Syncfusion.EJ2.Maps.MapsColorMapping> {
        new MapsColorMapping {Value="Permanent", Color="#D84444"},
        new MapsColorMapping {Value="Non-Permanent", Color="#316DB5"}
    };
}
@Html.EJS().Maps("maps").Format("c").UseGroupingSeparator(true).Layers(l =>
{
    l.ShapeSettings(s =>
s.ColorValuePath("Membership").Fill("#E5E5E5").ColorMapping(colorMapping)).
    TooltipSettings(tooltip =>
tooltip.Visible(true).ValuePath("population")).

DataSource(internData).ShapeDataPath("Country").ShapePropertyPath("name").
    ShapeData(ViewBag.worldMap).Add();
}).Render()
```

INTERNATIONALIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Localization

The localization library allows localizing the default text content of the Maps component. The Maps component has the static text of some features such as tooltip of zoom toolbar, and that can be changed to any other culture(Arabic, Deutsch, French, etc) by defining the locale value and translation object.

<!-- markdownlint-disable MD033 -->

| Locale key words | Text to display |
|------------------|-----------------|
| Zoom | Zoom |
| ZoomIn | Zoom In |
| ZoomOut | Zoom Out |
| Reset | Reset |
| Pan | Pan |

To load translation object in the application, use `load` function of **L10n** class. For more information about localization, refer [here](#).

CSHTML


```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@Html.EJS().Maps("maps").Locale("ar-AR").Layers(l => {
    l.ShapeData(ViewBag.usmap).Add();
}).ZoomSettings(zs=>zs.Enable(true)).Render()
<script>
    ej.base.L10n.load({
        'ar-AR': {
            'maps': {
                ZoomIn: 'تكبير',
                ZoomOut: 'تصغير',
                Zoom: 'زوم',
                Pan: 'مقللة',
                Reset: 'إعادة تعيين'
            }
        }
    });
</script>

```

LOCALIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.electionData = GetElectionData();
            return View();
        }
        public object GetWorldMap()
        {
            string allText =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/world_map.js");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetElectionData()
        {
            string text =
            System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/electiondata.js");
            return JsonConvert.DeserializeObject(text);
        }
    }
}

```

Methods in ASP.NET MVC Maps component

Methods

This section explains the methods used in the Maps component.

getMinMaxLatitudeLongitude

The `getMinMaxLatitudeLongitude` method returns the minimum and maximum latitude and longitude values of the Maps visible area. This method returns a `IMinMaxLatitudeLongitude` object that contains the Maps minimum and maximum latitude and longitude coordinates.

CSHTML

```
@{
    var data = new []
    {
        new { latitude= 22.572646, longitude= 88.363895 },
        new { latitude= 25.0700428, longitude= 67.2847875 }
    };
}

< button id = "button" > GetMinMaxLatitudeLongitude</button >
<p id="coordinatesDisplay"></p>
@(Html.EJS().Maps("maps").Width("450px").ZoomSettings(zoom =>
zoom.ZoomFactor(7).Enable(true)).CenterPosition(center =>
center.Latitude(21.815447).Longitude(80.1932)).Layers(layers => {
    layers.MarkerSettings(marker => {

marker.Visible(true).AnimationDuration(1500).Shape(MarkerType.Circle).DataSource(data).Height(25).Width(25).Add();
    }).ShapeData(ViewBag.worldMap).Add();
}).Render())
<script>
    function formatKey(key) {
    if (key === 'minLatitude') {
        return 'Minimum Latitude';
    } else if (key === 'maxLatitude') {
        return 'Maximum Latitude';
    } else if (key === 'minLongitude') {
        return 'Minimum Longitude';
    } else if (key === 'maxLongitude') {
        return 'Maximum Longitude';
    }
    }
}
document.getElementById('button').onclick = () => {
    var maps = document.getElementById('maps').ej2_instances[0];
    mapBoundCoordinates = maps.getMinMaxLatitudeLongitude();
    const displayDiv = document.getElementById('coordinatesDisplay');
    displayDiv.innerHTML = '';
    if (mapBoundCoordinates) {
        for (const key in mapBoundCoordinates) {
            if (Object.hasOwnProperty.call(mapBoundCoordinates, key)) {
                const p = document.createElement('p');
                const formattedKey = formatKey(key);
                p.textContent = `${formattedKey}: ${mapBoundCoordinates[key]}`;
                displayDiv.appendChild(p);
            }
        }
    }
} else {
```

```
        displayDiv.textContent = 'No coordinates available';  
    }  
};  
</script>
```

GETMINMAX.CS

```
using System;  
using System.Collections.Generic;  
using System.Diagnostics;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using EJ2_Core_Application.Models;  
using Newtonsoft.Json;  
namespace EJ2_Core_Application.Controllers  
{  
    public class HomeController : Controller  
    {  
        public IActionResult Index()  
        {  
            ViewBag.world_map = GetWorldMap();  
            ViewBag.worldMap = GetMap();  
            return View();  
        }  
  
        // To access the data in Core  
        public object GetWorldMap()  
        {  
            string allText =  
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.js");  
            return JsonConvert.DeserializeObject(allText);  
        }  
  
        // To access the data in MVC  
        public object GetMap()  
        {  
            string allText =  
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));  
            return JsonConvert.DeserializeObject(allText, typeof(object));  
        }  
    }  
}
```

Minimum Latitude: 52.60145907226985

Maximum Latitude: 10.580907935909977

Minimum Longitude: 54.47891428571432

Maximum Longitude: 105.90748571428577



<!-- markdownlint-disable MD038 -->

Migration from Essential JS 1

This article describes the API migration process of Maps component from Essential JS 1 to Essential JS 2.

Size Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| **Height** | Not Applicable | **Property:**

`height`
`@Html.EJS().Maps("container").Load("load").Height('300px').Render()`

| Width | Not Applicable | **Property:** *width*

 @Html.EJS().Maps("container")
.Load("load")
.Width('400px').Render()|

Title and Subtitle Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Title Text | Not Applicable | **Property:** *title.text*

 @Html.EJS().Maps("container")
.Load("load").TitleSettings
(new
 Syncfusion.EJ2.Maps.MapsTitleSettings
{Text="Members of the UN Security
 Council"}).Render()|

| Subtitle Text | Not Applicable | **Property:** *title.subtitle.text*

 @Html.EJS().Maps("container")
.Load("load")
.TitleSettings(title
 =>SubtitleSettings
(new MapsSubTitleSettings
{Text="- In 2017"})).Render()|

| Title Alignment | Not Applicable | **Property:** *title.alignment*

 @Html.EJS().Maps("container")
.Load("load").TitleSettings
(new
 Syncfusion.EJ2.Maps.MapsTitleSettings
{Text="Members of the UN Security
 Council",
Alignment('Far')}).Render()|

| Subtitle Alignment | Not Applicable | **Property:** *title.subtitle.alignment*

 @Html.EJS().Maps("container")
.Load("load")
.TitleSettings(title
 =>SubtitleSettings
(new MapsSubTitleSettings
{Text="- In 2017",
Alignment='Far'
 })).Render()|

Layer Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Type | Not Applicable | **Property:** *layers.type*

 @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.Type('Layer')}).Render()|

| Layer Type | **Property:** *layers.layerType*

@Html.EJ().Map("container").Layers(lr
 =>{lr.LayerType(LayerType.Geometry)}))| **Property:**
layers.layerType

@Html.EJS().Maps("container").Load("load").Layers(lr
 =>{lr.LayerType('Geometry')}).Render()|

| Visible | Not Applicable | **Property:** *layers.visible*

 @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.Visible(true)}).Render()|

| Bing Map Type | **Property:** *layers.bingMapType*

 @Html.EJ().Map("container").Layers(lr
 =>{lr.LayerType(Syncfusion.JavaScript.DataVisualization.Models.LayerType.Bing)}))| **Property:**
layers.bingMapType

 @Html.EJS().Maps("container").Load("load").Layers(lr
 =>{lr.BingMapType('Aerial')}).Render()|

| Bing Map Key | **Property:** *layers.key*

 @Html.EJ().Map("container").Layers(lr
 =>{lr.Key("")})|

| **Property:** *layers.key*

 @Html.EJS().Maps("container").Load("load").Layers(lr
 =>{lr.Key("")}).Render()|

| URL Template | **Property:** *layers.urlTemplate* @Html.EJ().Map("container").Layers(lr =>{lr.UrlTemplate:'http://a.tile.openstreetmap.org/level/tileX/tileY.png' })| **Property:** *layers.urlTemplate* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.UrlTemplate:'http://a.tile.openstreetmap.org/level/tileX/tileY.png' }).Render()|

| Shape Data | **Property:** *layers.shapeData* @Html.EJ().Map("container").Layers(lr =>{lr.ShapeData(usmap).Add}))| **Property:** *layers.shapeData* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.ShapeData('usmap')}).Render()|

| Data Source | **Property:** *layers.dataSource* @Html.EJ().Map("container").Layers(lr =>{lr.dataSource('PopulationData')})| **Property:** *layers.dataSource* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.DataSource("")}).Render()|

| Query | Not Applicable | **Property:** *layers.query* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.Query("")}).Render()|

| Shape Data Path | **Property:** *layers.shapeDataPath* @Html.EJ().Map("container").Layers(lr =>{lr.shapeDataPath('population')})| **Property:** *layers.shapeDataPath* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.ShapeDataPath("")}).Render()|

| Shape Property Path | **Property:** *layers.shapePropertyPath* @Html.EJ().Map("container").Layers(lr =>{lr.ShapePropertyPath('Continent')})| **Property:** *layers.shapePropertyPath* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.shapePropertyPath("")}).Render()|

| Layer Animation | Not Applicable | **Property:** *layers.animationDuration* @Html.EJS().Maps("container").Load("load").Layers(lr =>{lr.ShapePropertyPath("")}).Render()|

Shape Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Shape Fill | **Property:** *layers.shapeSettings.fill* @Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(ss =>{ss.Fill('#626171')})})| **Property:** *layers.shapeSettings.fill* @Html.EJS().Maps("container").Load("load").ShapeSettings(new MapsShapeSettings{Fill="red"}).Render()|

| Shape Palette | **Property:** *layers.shapeSettings.colorPalette* @Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(ss =>{ss.ColorPalette(ColorPalette.CustomPalette)})}) .CustomPalette(new List<string>{"#E51400", "#A4C400", "#730202", })| **Property:** *layers.shapeSettings.palette* @Html.EJS().Maps("container").Load("load").ShapeSettings(new MapsShapeSettings{ColorMapping = ViewBag.colorMappings}).Render() ViewBag.colorMappings = data;|

| Shape Point Radius | Not Applicable | **Property:** *layers.shapeSettings.circleRadius* @Html.EJS().Maps("container").Load("load").ShapeSettings(new MapsShapeSettings{circleRadius= 10 }).Render()|

| Shape Color Value Path | **Property:** *layers.shapeSettings.colorValuePath*

 @(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(ss =>{ss.ColorValuePath("")}})) |
Property: *layers.shapeSettings.colorValuePath*

 @Html.EJS().Maps("container").Load("load").ShapeSettings(new
 MapsShapeSettings{ColorValuePath=""}.Render() |

| Shape Value Path | **Property:** *layers.shapeSettings.valuePath*

 @(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(ss =>{ss.ValuePath("")}})) | **Property:**
layers.shapeSettings.valuePath

 @Html.EJS().Maps("container").Load("load").ShapeSettings(new MapsShapeSettings{
 valuePath='population'}).Render() |

| Shape DashArray | Not Applicable | **Property:** *layers.shapeSettings.dashArray*

 @Html.EJS().Maps("container").Load("load").ShapeSettings(new
 MapsShapeSettings{DashArray='1,2'}).Render() |

| Shape Opacity | Not Applicable | **Property:** *layers.shapeSettings.opacity*

 @Html.EJS().Maps("container").Load("load").ShapeSettings(new MapsShapeSettings{
 Opacity='0.5'}).Render() |

| Range Color Mapping | **Property:** *layers.shapeSettings.colorMappings.rangeColorMapping*

 @(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(ss =>{ss..RangeColorMappings(cm
 =>cm.From(10).To(100).color('blue')}})) | **Property:** *layers.shapeSettings.colorMapping*

 @Html.EJS().Maps("container").Load("load").ShapeSettings(new
 MapsShapeSettings{ColorMapping = ViewBag.ColorMapping
 }).Render()
ViewBag.ColorMapping = colorMapping;
colorMapping.Add(new
 MapsColorMapping{From =100, To=1000, Color = "#b3daff", Label = null}) |

| Equal Color Mapping | **Property:** *layers.shapeSettings.colorMappings.equalColorMapping*

 @(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(ss =>{ss.EqualColorMappings(cm
 =>cm.value("").color('blue')}})) | **Property:** *layers.shapeSettings.colorMapping*

 @Html.EJS().Maps("container").Load("load").ShapeSettings(new
 MapsShapeSettings{ColorMapping = ViewBag.ColorMapping
 }).Render()
ViewBag.ColorMapping = colorMapping;
colorMapping.Add(new
 MapsColorMapping{Value = "1", Color = "#b3daff", Label = null}) |

Marker Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Marker Data Source | **Property:** *layers.markers*

@(Html.EJ().Map("container").Layers(lr
 =>{lr.Markers(datasource)}))
 ViewData["datasource"]=
 Syncfusion_LocationData.GetSyncfusionLocationData();
List<MapMarker>
 syncfusionLocationData = new List<MapMarker>{new LocationData {Name = "USA", Latitude
 =38.8833 , Longitude = -77.0167 }} | **Property:** *layers.markerSettings.dataSource*

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
 args.maps.layers[0].markerSettings=[{dataSource:[{latitude: 37.6276571, longitude: -
 122.4276688, name: 'San Bruno' },]} |

| Marker Template | **Property:** *layers.markerTemplate*

@Html.EJS().Map("container").Layers(lr
=>{lr.Markers(datasource)..MarkerTemplate("template")}) | **Property:**
layers.markerSettings.template

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{template:'Template'}] |

| Marker Visible | Not Applicable | **Property:** *layers.markerSettings.visible*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{visible:true}] |

| Marker Fill | Not Applicable | **Property:** *layers.markerSettings.fill*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{fill:'red'}] |

| Marker Height | Not Applicable | **Property:** *layers.markerSettings.height*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{height:20}] |

| Marker Width | Not Applicable | **Property:** *layers.markerSettings.width*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{width:20}] |

| Marker Shape | Not Applicable | **Property:** *layers.markerSettings.shape*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{shape="Balloon"}] |

| Marker ImageURL | Not Applicable | **Property:** *layers.markerSettings.imageUrl*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{imageUrl:''}] |

| Marker Opacity | Not Applicable | **Property:** *layers.markerSettings.opacity*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{opacity:0.5}] |

| Marker Legend Text | Not Applicable | **Property:** *layers.markerSettings.legendText*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{legendText:'China'}] |

| Marker Offset | Not Applicable | **Property:** *layers.markerSettings.offset*

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{offset:new Point(20, 20)}] |

| Marker Animation Duration | Not Applicable | **Property:**
layers.markerSettings.animationDuration

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings = [{animationDuration:2000}] |

| Marker Animation Delay | Not Applicable | **Property:**
layers.markerSettings.animationDelay

@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings =[{animationDelay:100}]|

| Marker DashArray | Not Applicable | **Property:** *layers.markerSettings.dashArray*
@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings =[{dashArray:'2,3'}]|

| Marker Selection | Not Applicable | **Property:** *layers.markerSettings.selectionSettings*
@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings =[{ selectionSettings : {
enable:true,fill:'#D2691E',opacity:1,enableMultiSelect:false }}]|

| Marker Highlight | Not Applicable | **Property:** *layers.markerSettings.highlightSettings*
@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings =[{ highlightSettings : { enable:true,fill:'#D2691E',opacity:1
}}]|

| Marker Tooltip | Not Applicable | **Property:** *layers.markerSettings.tooltipSettings*
@Html.EJS().Maps("container").Load("mapsLoad").Render()
function mapsLoad(args){
args.maps.layers[0].markerSettings =[{tooltipSettings : {
visible:true,fill:'#363F4C',template:'TooltipTemplate', valuePath:'State',
format:'\${State}\${District}' }}]|

Bubble Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | **Property:** *layers.bubbleSettings.visible*
@Html.EJ().Map("container").Layers(lr
=>{lr.BubbleSettings(bs =>{bs.ShowBubble(true) })})| **Property:**
layers.bubbleSettings.visible
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr
=>{lr.BubbleSettings(ViewBag.bubbleSettings)}).Render()
MapsBubble bubble = new
MapsBubble();
bubble.Visible=true|

| ValuePath | **Property:**
layers.bubbleSettings.valuePath
@Html.EJ().Map("container").Layers(lr
=>{lr.BubbleSettings(bs =>{bs.ValuePath('Population') })})| **Property:**
layers.bubbleSettings.valuePath
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubble
Settings)}).Render()
MapsBubble bubble = new
MapsBubble();
bubble.ValuePath='Population'|

| MinValue | **Property:** *layers.bubbleSettings.minValue*
@Html.EJ().Map("container").Layers(lr =>{lr.BubbleSettings(bs =>{bs.MinValue(20) })})|
Property:
layers.bubbleSettings.minRadius
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings)}).Render()
MapsBubble bubble = new
MapsBubble();
bubble.MinRadius=10|

| MaxValue | **Property:**

```
layers.bubbleSettings.maxValue<br/><br/>@(Html.EJ().Map("container").Layers(lr
=>{lr.BubbleSettings(bs =>{bs.MaxValue(30) }))))
```

| **Property:** layers.bubbleSettings.maxRadius


```
@Html.EJS().Maps("container").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings))).Render(
)<br/><br/>MapsBubble bubble = new MapsBubble();bubble.MaxRadius=20|
```

| Bubble Type | Not Applicable | **Property:**

```
layers.bubbleSettings.bubbleType<br/><br/>@(Html.EJS().Maps("container").Load("mapsLoad").Lay
er(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings))).Render()<br/>MapsBubble bubble = new
MapsBubble();<br/>bubble.BubbleType='Circle'|
```

| Color | **Property:** layers.bubbleSettings.color

 @(Html.EJ().Map("container").Layers(lr

```
=>{lr.BubbleSettings(bs =>{bs.Color('red') })))) | Property: layers.bubbleSettings.fill<br/><br/>
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings))).Render()<br/>MapsBubble bubble = new MapsBubble();<br/>bubble.Fill='red'|
```

| Opacity | **Property:**

```
layers.bubbleSettings.bubbleOpacity<br/><br/>@(Html.EJ().Map("container").Layers(lr
=>{lr.BubbleSettings(bs =>{bs.BubbleOpacity(0.6) })))) | Property:
layers.bubbleSettings.opacity<br/><br/>
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{layer.BubbleSettings(ViewBag.bubbleSettings))).Render()<br/>MapsBubble bubble = new
MapsBubble();<br/>bubble.Opacity=0.5|
```

| Color Value Path | **Property:** layers.bubbleSettings.colorValuePath


```
@(Html.EJ().Map("container").Layers(lr =>{lr.BubbleSettings(bs =>{bs.ColorValuePath("")})))) |
Property: layers.bubbleSettings.colorValuePath<br/><br/>
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{layer.BubbleSettings(ViewBag.bubbleSettings))).Render()<br/>MapsBubble bubble = new
MapsBubble();<br/>bubble.ColorValuePath=""|
```

| Enable Tooltip | **Property:**

```
layers.bubbleSettings.showTooltip<br/><br/>@(Html.EJ().Map("container").Layers(lr
=>{lr.BubbleSettings(bs =>{bs.ShowTooltip(true) })))) | Property:
layers.bubbleSettings.tooltipSettings.visible<br/><br/>
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings))).Render()<br/>MapsBubble bubble = new
MapsBubble();<br/>bubble.TooltipSetting.Visible=true|
```

| Tooltip Template | **Property:** layers.bubbleSettings.tooltipTemplate


```
@(Html.EJ().Map("container").Layers(lr =>{lr.BubbleSettings(bs
=>{bs.TooltipTemplate('Template') })))) | Property:
layers.bubbleSettings.tooltipSettings.template<br/><br/>
@Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings))).Render()<br/>MapsBubble bubble = new
MapsBubble();<br/>bubble.TooltipSetting.Template='template'|
```

| Bubble Selection | Not Applicable | **Property:** *layers.bubbleSettings.selectionSettings*

 @(Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{layer.BubbleSettings(ViewBag.bubbleSettings)}).Render())
MapsBubble bubble = new
 MapsBubble();
bubble.SelectionSetting.Enable=true |

| Bubble Highlight | Not Applicable | **Property:** *layers.bubbleSettings.highlightSettings*

 @(Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings)}).Render())
MapsBubble bubble = new
 MapsBubble();
bubble.HighlightSetting.Enable=true |

| Range Color Mapping | **Property:** *layers.bubbleSettings.colorMappings.rangeColorMapping*

 @(Html.EJ().Map("container").Layers(lr =>{lr.BubbleSettings(bs =>{bs. })})) | **Property:**
layers.bubbleSettings.colorMapping

@(Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{lr.BubbleSettings(ViewBag.bubbleSettings)}).Render())
MapsBubble bubble = new
 MapsBubble();
bubble. |

| Equal Color Mapping | **Property:** *layers.bubbleSettings.colorMappings.equalColorMapping*

 @(Html.EJ().Map("container").Layers(lr =>{lr.BubbleSettings(bs =>{bs. })})) | **Property:**
layers.bubbleSettings.colorMapping

 @(Html.EJS().Maps("container").Load("mapsLoad").Layer(lr=>{layer.BubbleSettings(ViewBag.bubbleSettings)}).Render())
MapsBubble bubble = new MapsBubble();
bubble. |

DataLabel Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | **Property:** *layers.labelSettings.showLabels*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LabelSettings(lb =>{lb.ShowLabels(true)}))}) |
Property: *layers.dataLabelSettings.visible*

 @(Html.EJS().Maps("container").Layer({lr.DataLabelSettings(new MapsDataLabelSettings{Visible
 = true}))}.Render()) |

| Label Path | **Property:**
layers.labelSettings.labelPath

@(Html.EJ().Map("container").Layers(lr
 =>{lr.LabelSettings(lb =>{lb.LabelPath("iso31662")}))}) | **Property:**
layers.dataLabelSettings.labelPath

 @(Html.EJS().Maps("container").Layer({lr.DataLabelSettings(new
 MapsDataLabelSettings{LabelPath = "name",}))}.Render()) |

| Enable Smart Label | **Property:** *layers.labelSettings.enableSmartLabel*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LabelSettings(lb
 =>{lb.EnableSmartLabel(true)}))}) | Not Applicable |

| Smart Label Size | **Property:** *layers.labelSettings.smartLabelSize*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LabelSettings(lb =>{lb.SmartLabelSize(20)}))}) | Not
 Applicable |

| Label Length | **Property:** *layers.labelSettings.labelLength*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LabelSettings(lb =>{lb.LabelLength(20)}))}) | Not
 Applicable |

| Opacity | Not Applicable | **Property:** *layers.dataLabelSettings.opacity*

 @Html.EJS().Maps("container").Layer({lr.DataLabelSettings(new
 MapsDataLabelSettings{Opacity=0.5})).Render() |

| Smart Label Mode | Not Applicable | **Property:** *layers.dataLabelSettings.smartLabelMode*

 @Html.EJS().Maps("container").Layer({lr.DataLabelSettings(new MapsDataLabelSettings{
 SmartLabelMode = SmartLabelMode.Trim})).Render() |

| IntersectAction | Not Applicable | **Property:** *layers.dataLabelSettings.intersectionAction*

 @Html.EJS().Maps("container").Layer({lr.DataLabelSettings(new MapsDataLabelSettings{
 IntersectionAction = IntersectionAction.Trim})).Render() |

| Template | Not Applicable | **Property:** *layers.dataLabelSettings.template*

 @Html.EJS().Maps("container").Layer({lr.DataLabelSettings(new
 MapsDataLabelSettings{Template='Temaplate'})).Render() |

Legend Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | **Property:** *layers.legendSettings.showLegend*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(ls =>{ls.ShowLegend(true)}})) |
Property: *legendSettings.visible*

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{ Visible = true}).Render() |

| Toggle Visibility | **Property:**
layers.legendSettings.toggleVisibility

 @(Html.EJ().Map("container").Layers(lr
 =>{lr.LegendSettings(ls =>{ls.ToggleVisibility(true)}})) | **Property:**
legendSettings.toggleVisibility

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{ToggleVisibility = true}).Render() |

| Legend Location X | **Property:** *layers.legendSettings.positionX*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(ls =>{ls.PositionX(250)}})) |
Property: *legendSettings.location*

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{ }).Render() |

| Legend Location Y | **Property:** *layers.legendSettings.positionY*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(ls =>{ls.PositionY(250)}})) |
Property: *legendSettings.location*

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{ }).Render() |

| Legend Type | **Property:** *layers.legendSettings.type*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(ls =>{ls.Type:'Layers'}})) | **Property:**
legendSettings.type

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{ Type =
 Syncfusion.EJ2.Maps.LegendType.Layers}).Render() |

| Label Orientation | **Property:** *layers.legendSettings.labelOrientation*

 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(ls
 =>{ls.LabelOrientation('Vertical')}})) | Not Applicable |

| Legend Title | **Property:** *layers.legendSettings.title*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.Title:'Union territories of India'}})}) | **Property:** *legendSettings.title*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ Title= ""}).Render() |

| Legend Mode | **Property:** *layers.legendSettings.mode*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.Mode('Default')}})}) | **Property:** *legendSettings.mode*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ Mode = Syncfusion.EJ2.Maps.LegendMode.Default, }).Render() |

| Legend Position | **Property:** *layers.legendSettings.position*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.Position('TopLeft')}})}) | **Property:** *legendSettings.position*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ Position = Syncfusion.EJ2.Maps.LegendPosition.Bottom, }).Render() |

| Legend DockOnMap | **Property:** *layers.legendSettings.dockOnMap*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.DockOnMap(true)}})}) | Not Applicable |

| Legend Alignment | **Property:** *layers.legendSettings.dockPosition*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.DockPosition('Right')}})}) | **Property:** *legendSettings.alignment*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Alignment = Syncfusion.EJ2.Maps.Alignment.Center }).Render() |

| Legend Left Label | **Property:** *layers.legendSettings.leftLabel*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.LeftLabel('1000M')}})}) | Not Applicable |

| Legend Right Label | **Property:** *layers.legendSettings.rightLabel*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.RightLabel('3000M')}})}) | Not Applicable |

| Legend Shape | **Property:** *layers.legendSettings.icon*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.Icon('Circle')}})}) | **Property:** *legendSettings.shape*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Shape= Syncfusion.EJ2.Maps.LegendShape.Circle }).Render() |

| Legend Shape Height | **Property:** *layers.legendSettings.iconHeight*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.IconHeight(20)}})}) | **Property:** *legendSettings.shapeHeight*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ShapeHeight=20 }).Render() |

| Legend Shape Width | **Property:** *layers.legendSettings.iconWidth*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(Is =>{Is.IconWidth(20)}})}) |

Property:

legendSettings.shapeWidth
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ShapeWidth=20 }).Render() |

| Height | **Property:** *layers.legendSettings.height*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(lr =>{lr.Height(50)}))}) | **Property:** *legendSettings.width*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Height='50'}).Render() |

| Width | **Property:** *layers.legendSettings.width*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(lr =>{lr.Width(50)}))}) | **Property:** *legendSettings.width*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Width='150' }).Render() |

| Show Labels | **Property:** *layers.legendSettings.showLabels*
 @(Html.EJ().Map("container").Layers(lr =>{lr.LegendSettings(lr =>{lr.ShowLabels(true)}))}) | Not Applicable |

| Background | Not Applicable | **Property:** *legendSettings.background*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Background='transparent' }).Render() |

| Label Position | Not Applicable | **Property:** *legendSettings.labelPosition*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{LabelPosition = Syncfusion.EJ2.Maps.LabelPosition.After }).Render() |

| Label Display Mode | Not Applicable | **Property:** *legendSettings.labelDisplayMode*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{LabelDisplayMode = Syncfusion.EJ2.Maps.LabelIntersectAction.Trim, }).Render() |

| Legend Orientation | Not Applicable | **Property:** *legendSettings.orientation*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Orientation = Syncfusion.EJ2.Maps.LegendArrangement.Horizontal, }).Render() |

| Legend Item Fill | Not Applicable | **Property:** *legendSettings.fill*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{Fill='red'}).Render() |

| Legend Shape Padding | Not Applicable | **Property:** *legendSettings.shapePadding*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ShapePadding=20}).Render() |

| Legend Shape Border Color | Not Applicable | **Property:** *legendSettings.shapeBorder.color*
 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{ShapeBorder = { Color="green" }, }).Render() |

| Legend Shape Border Width | Not Applicable | **Property:** *legendSettings.shapeBorder.width*

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{ShapeBorder = { Width=30 },}).Render() |

| Inverter Pointer | Not Applicable | **Property:** *legendSettings.invertedPointer*

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{InvertedPointer= true }).Render() |

| Item Text Style | Not Applicable | **Property:** *legendSettings.textStyle*

 @Html.EJS().Maps("container").LegendSettings(new
 Syncfusion.EJ2.Maps.MapsLegendSettings{TextStyle = new MapsFont { Size = "14px" }
 }).Render() |

| Title Style | Not Applicable | **Property:** *legendSettings.textStyle*

 @Html.EJS().Maps("container").LegendSettings(new Syncfusion.EJ2.Maps.MapsLegendSettings{
 TitleStyle= new MapsFont { Size = "14px" } }).Render() |

Zooming Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable | Not Applicable | **Property:** *zoomSettings.enableZoom*

 @Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{
 Enable=true}).Render() |

| Minimum Zoom | **Property:**
zoomSettings.minValue

@Html.EJ().Map("container").ZoomSettings(zm=>{zm.MinValu
 e(2)})) | **Property:** *zoomSettings.minZoom*

 @Html.EJS().Maps("container").ZoomSettings(new
 Syncfusion.EJ2.Maps.MapsZoomSettings{MinZoom=2,}).Render() |

| Maximum Zoom | **Property:** *zoomSettings.maxValue*

 @Html.EJ().Map("container").ZoomSettings(zm=>{zm.MaxValue(50)})) | **Property:**
zoomSettings.maxZoom

 @Html.EJS().Maps("container").ZoomSettings(new
 Syncfusion.EJ2.Maps.MapsZoomSettings{MaxZoom=4,}).Render() |

| Mouse Wheel Zoom | **Property:** *zoomSettings.enableMouseWheelZoom*

 @Html.EJ().Map("container").ZoomSettings(zm=>{zm.EnableMouseWheelZoom(true)})) |
Property: *zoomSettings.maxZoom*

 @Html.EJS().Maps("container").ZoomSettings(new
 Syncfusion.EJ2.Maps.MapsZoomSettings{MouseWheelZoom=true}).Render() |

| Double Click Zoom | Not Applicable | **Property:** *zoomSettings.doubleClickZoom*

 @Html.EJS().Maps("container").ZoomSettings(new
 Syncfusion.EJ2.Maps.MapsZoomSettings{DoubleClickZoom=true}).Render() |

| Pinch Zoom | Not Applicable | **Property:**
zoomSettings.pinchZooming

@Html.EJS().Maps("container").ZoomSettings(new
 Syncfusion.EJ2.Maps.MapsZoomSettings{PinchZooming=true}).Render() |

| Single Click Zoom | **Property:** *zoomSettings.enableZoomOnSelection*

 @Html.EJ().Map("container").ZoomSettings(zm=>{zm.EnableZoomOnSelection(true)})) |

Property: `zoomSettings.zoomOnClick`
`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{ZoomOnClick=true}).Render()`

| Zoom Factor | **Property:** `zoomSettings.factor`

`@(Html.EJ().Map("container").ZoomSettings(zm=>{zm.Factor(2)}))` | **Property:** `zoomSettings.zoomFactor`
`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{ ZoomFactor=2}).Render()`

| Toolbars | Not Applicable | **Property:** `zoomSettings.toolbars`

`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{Toolbars =['ZoomIn']}).Render()`

| Toolbar Orientation | Not Applicable | **Property:** `zoomSettings.toolBarOrientation`

`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{ToolBarOrientation= Syncfusion.EJ2.Maps.Orientation.Horizontal}).Render()`

| Toolbar Vertical Alignment | Not Applicable | **Property:** `zoomSettings.verticalAlignment`

`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{VerticalAlignment = Syncfusion.EJ2.Maps.Alignment.Center}).Render()`

| Toolbar Horizontal Alignment | Not Applicable | **Property:**

`zoomSettings.horizontalAlignment`
`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{HorizontalAlignment=Syncfusion.EJ2.Maps.Alignment.Center}).Render()`

| Toolbar Highlight Color | Not Applicable | **Property:** `zoomSettings.highlightColor`

`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{HighlightColor="#e61576"}).Render()`

| Toolbar Selection Color | Not Applicable | **Property:** `zoomSettings.selectionColor`

`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{SelectionColor="#e61576"}).Render()`

| Toolbar Fill Color | Not Applicable | **Property:** `zoomSettings.color`

`@Html.EJS().Maps("container").ZoomSettings(new Syncfusion.EJ2.Maps.MapsZoomSettings{Color="#e61576"}).Render()`

Highlight And Selection Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Highlight Fill | **Property:** `layers.shapeSettings.highlightColor`

`@(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(sp =>{sp.HighlightColor('green')})}))` | **Property:** `fill`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{HighlightSettings = new MapsHighlightSettings{Fill='red'}}).Render()`

| Enable Highlight | **Property:**

`layers.enableMouseHover`
`@(Html.EJ().Map("container").Layers(lr`


```

=>{lr.EnableMouseHover(true)}})| Property: enable<br/><br/>
@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{HighlightSettings = new
MapsHighlightSettings{Enable=true}}).Render()|

| Highlight Border Color | Property: layers.shapeSettings.highlightStroke<br/><br/>
@(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(sp =>{sp.HighlightStroke('red')}}))}|
Property: layers.highlightSettings.border.color<br/><br/>
@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{HighlightSettings = new
MapsHighlightSettings{Border={Color='green'}}}).Render()|

| Highlight Border Width | Property: layers.shapeSettings.highlightBorderWidth<br/><br/>
@(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(sp
=>{sp.HighlightBorderWidth(2)}}))}| Property: layers.highlightSettings.border.width<br/><br/>
@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{HighlightSettings = new
MapsHighlightSettings{Border={Width=2}}}).Render()|

| Highlight Opacity | Not Applicable | Property: layers.layers.highlightSettings.opacity<br/><br/>
@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{HighlightSettings = new
MapsHighlightSettings{Opacity=0.3}}).Render()|

| Selection Fill | Property:
layers.shapeSettings.selectionColor<br/><br/>@(Html.EJ().Map("container").Layers(lr
=>{lr.ShapeSettings(sp =>{sp.SelectionColor('blue')}}))}| Property:
layers.selectionSettings.fill<br/><br/> @Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{SelectionSettings = new
MapsSelectionSettings{Fill='red'}}).Render()|

| Selection Enable | Property:
layers.enableSelection<br/><br/>@(Html.EJ().Map("container").Layers(lr
=>{lr.EnableSelection(true)}})| Property: layers.selectionSettings.enable<br/><br/>
@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{SelectionSettings = new MapsSelectionSettings{Enable=
true}}).Render()|

| Selection Border Width | Property: layers.selectionSettings.selectionStrokeWidth<br/><br/>
@(Html.EJ().Map("container").Layers(lr =>{lr.ShapeSettings(sp
=>{sp.SelectionStrokeWidth(2)}}))}| Property: layers.selectionSettings.border.width<br/><br/>
@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{SelectionSettings = new
MapsSelectionSettings{Border={Width=2}}}).Render()|

| Selection Border Color | Property:
layers.selectionSettings.selectionStroke<br/><br/>@(Html.EJ().Map("container").Layers(lr
=>{lr.ShapeSettings(sp =>{sp.SelectionStroke("white')}}))}| Property:
layers.selectionSettings.border.color<br/><br/> @Html.EJS().Maps("container").Layers(new

```

```
List<Syncfusion.EJ2.Maps.MapsLayer>{SelectionSettings = new
MapsSelectionSettings{Border={Color='red'}}}.Render()|
```

| Selection Opacity | Not Applicable | **Property:** *layers.selectionSettings.opacity*

@Html.EJS().Maps("container").Layers(new
List<Syncfusion.EJ2.Maps.MapsLayer>{SelectionSettings = new
MapsSelectionSettings{Opacity=0.3}}).Render()|

Navigation Line Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Visible | Not Applicable | **Property:** *layers.navigationLineSettings.visible*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{ visible: true}}|

| Width | Not Applicable | **Property:** *layers.navigationLineSettings.width*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{width:2}}|

| Longitude | Not Applicable | **Property:** *layers.navigationLineSettings.longitude*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{longitude: [-97.8717041015625, -
89.6649169921875]}}|

| Latitude | Not Applicable | **Property:** *layers.navigationLineSettings.latitude*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{Latitude= [22.403410892712124,
21.282336521195344] }}|

| DashArray | Not Applicable | **Property:**
layers.navigationLineSettings.dashArray

@Html.EJS().Maps("container").Render()
fu
nction mapsLoad(args){args.maps.layers[0].navigationLineSettings{dashArray:"1,2"}}|

| Color | Not Applicable | **Property:** *layers.navigationLineSettings.color*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{color:"green"}}|

| Angle | Not Applicable | **Property:** *layers.navigationLineSettings.angle*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{angle:180}}|

| Arrow Position | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.position*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{arrow:{ position:"start" }}}|

| Show Arrow | Not Applicable | **Property:** *layers.navigationLineSettings.arrow.showArrow*

@Html.EJS().Maps("container").Render()
function
mapsLoad(args){args.maps.layers[0].navigationLineSettings{arrow:{ showArrow:true }}}|

| Arrow size | Not Applicable | **Property:** `layers.navigationLineSettings.arrow.size`
`@Html.EJS().Maps("container").Render()`
`function mapsLoad(args){args.maps.layers[0].navigationLineSettings{ arrow:{ size:2 }}}|`

| Arrow Color | Not Applicable | **Property:** `layers.navigationLineSettings.arrow.color`
`@Html.EJS().Maps("container").Render()`
`function mapsLoad(args){args.maps.layers[0].navigationLineSettings{arrow:{ color:'red' }}}|`

| Arrow Offset | Not Applicable | **Property:** `layers.navigationLineSettings.arrow.offSet`
`@Html.EJS().Maps("container").Render()`
`function mapsLoad(args){args.maps.layers[0].navigationLineSettings{arrow:{ offSet:10 }}}|`

Tooltip Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Tooltip Enable | **Property:** `layers.showTooltip`
`@(Html.EJ().Map("container").Layers(lr =>{lr.ShowTooltip(true)}))` | **Property:** `layers.tooltipSettings.visible`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{Visible= true }}}).Render()`

| Tooltip Template | **Property:** `layers.tooltipTemplate`
`@(Html.EJ().Map("container").Layers(lr =>{lr.ToolTipTemplate('myTooltip').Add}))` | **Property:** `layers.tooltipSettings.visible`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{Template='tremplate' }}}).Render()`

| Value Path | Not Applicable | **Property:** `layers.tooltipSettings.valuePath`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{ValuePath = "State" }}}).Render()`

| Format | Not Applicable | **Property:** `layers.tooltipSettings.format`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{Format='${State}' }}}).Render()`

| Border Color | Not Applicable | **Property:** `layers.tooltipSettings.border.color`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{Border={ color:'red' } }}}).Render()`

| Border Width | Not Applicable | **Property:** `layers.tooltipSettings.border.width`
`@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{Border={ width:"" }}}).Render()`

| Text Style | Not Applicable | **Property:** *layers.tooltipSettings.textStyle*

 @Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>{ new
 Syncfusion.EJ2.Maps.MapsLayer{ TooltipSettings = new MapsTooltipSettings{ TextStyle= {
 Size="15px", Color="red", FontFamily="arial", FontWeight="bold", FontStyle="normal",
 Opacity=0.8 }}}}).Render()|

Annotation Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Content | Not Applicable | **Property:** *legendSettings.annotations.content*

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function
 mapsLoad(args){args.maps.annotations=[{ content:'USA Population 2018'}]}|

| Location X | Not Applicable | **Property:** *legendSettings.annotations.x*

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function
 mapsLoad(args){args.maps.annotations=[{ x:'250px' }]}|

| Location Y | Not Applicable | **Property:** *legendSettings.annotations.y*

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function
 mapsLoad(args){args.maps.annotations=[{ y:'150px' }]}|

| Vertical Alignment | Not Applicable | **Property:**
legendSettings.annotations.verticalAlignment

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function
 mapsLoad(args){args.maps.annotations=[{verticalAlignment:'Center'}]}|

| Horizontal Alignment | Not Applicable | **Property:**
legendSettings.annotations.horizontalAlignment

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function
 mapsLoad(args){args.maps.annotations=[{horizontalAlignment:'Center'}]}|

| Zindex | Not Applicable | **Property:** *legendSettings.annotations.zIndex*

 @Html.EJS().Maps("container").Load("mapsLoad").Render()
function
 mapsLoad(args){args.maps.annotations=[{ zIndex:'-1' }]}|

Maps Other Properties Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Projection Type | Not Applicable | **Property:** *projectionType*

 @Html.EJS().Maps("container").ProjectionType(ProjectionType.Eckert3).Render()|

| Background | **Property:** *background* @Html.EJ().Maps("container").Background('red'))|
Property: *background*

 @Html.EJS().Maps("container").Background("red").Render()|

| Enable Group Separator | **Property:** *enableGroupSeparator*

 @Html.EJ().Maps("container").UseGroupingSeparator(true))| **Property:**
useGroupingSeparator

 @Html.EJS().Maps("container").UseGroupingSeparator(true).Render()|

| Base Layer Index | **Property:** *baseMapIndex*

 @Html.EJ().Maps("container").BaseLayerIndex(1)| **Property:** *baseLayerIndex*

 @Html.EJS().Maps("container").BaseLayerIndex(1).Render()|

| locale | **Property:** *locale* @Html.EJ().Map("container").Locale('en-us') | Not Applicable |

| Responsive | **Property:** *isResponsive*

 @Html.EJ().Map("container").IsResponsivet(true)| Not Applicable |

| Enable Pan | **Property:** *enablePan* @Html.EJ().Map("container").EnablePan(true)| Not Applicable |

| Enable Navigation | **Property:** *navigationControl.enableNavigation*

 @Html.EJ().Map("container").NavigationControl(new{ enableNavigation=true })| Not Applicable |

| Navigation Orientation | **Property:** *navigationControl.orientation*

 @Html.EJ().Map("container").NavigationControl(new{ orientation='vertical' })| Not Applicable |

| Navigation Dock Position | **Property:** *navigationControl.dockPosition* @Html.EJ().Map("container").NavigationControl(new{ dockPosition='centerleft' })| Not Applicable |

| Navigation Absolute Position | **Property:** *navigationControl.absolutePosition*

 @Html.EJ().Map("container").NavigationControl(new{ absolutePosition={ x: 100, y : 100 } })| Not Applicable |

| Dragging Selection | **Property:** *draggingOnSelection*

 @Html.EJ().Map("container").DraggingOnSelection(true)| Not Applicable |

| Resize | **Property:** *enableResize* @Html.EJ().Map("container").EnableResize(true))| Not Applicable |

| Enable Animation | **Property:** *enableAnimation*

 @Html.EJ().Map("container").EnableAnimation(true))| Not Applicable |

| Enable Layer Animation | **Property:** *enableLayerChangeAnimation* @Html.EJ().Map("container").EnableLayerChangeAnimation(true))| Not Applicable |

| Center Position | **Property:** *centerPosition*

 @Html.EJS().Maps("container").CenterPosition(new { latitude = 35.65, longitude = -97.3 })|
Property: *centerPosition*

 @Html.EJS().Maps("container").CenterPosition(new { latitude = 35.65, longitude = -97.3 }).Render()|

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Shape Selected | **Property:** *shapeSelected*

@Html.EJ().Map("container").ShapeSelected("shapeSelected")
function shapeSelected(args){} | **Property:** *shapeSelected*

@Html.EJS().Maps("container").ShapeSelected("shapeSelected").Render()
shapeSelected(args){}

| Marker Selected | **Property:** *markerSelected*

@(Html.EJ().Map("container").MarkerClick("markerClick"))
Property: *markerClick*

@Html.EJS().Maps("container").Load("mapLoad").Render()
function mapLoad(args){}

| Marker Move | **Property:** *markerEnter* | **Property:** *markerMouseMove*

@(Html.EJ().Map("container").MarkerClick("markerClick"))
function markerClick(args){}

@Html.EJS().Maps("container").Load("mapLoad").Render()
function mapLoad(args){}

| Marker Leave | **Property:** *markerLeave*

@(Html.EJ().Map("container").MarkerLeave("markerLeave"))
markerLeave(args){}

| Legend Item Rendering | **Property:** *legendItemRendering*

@(Html.EJ().Map("container").LegendItemRendering("legendItemRendering"))
legendItemRendering(args){}

| Display Text Rendering | **Property:** *displayTextRendering*

@(Html.EJ().Map("container").DisplayTextRendering("displayTextRendering"))
displayTextRendering(args){}

Property: *dataLabelRendering*

@Html.EJS().Maps("container").DataLabelRendering("dataLabelRendering").Render()
dataLabelRendering(args){}

| Legend Item Click | **Property:** *legendItemClick*

@(Html.EJ().Map("container").LegendItemClick("legendItemClick"))
legendItemClick(args){}

| Bubble Rendering | **Property:** *bubbleRendering*

@(Html.EJ().Map("container").BubbleRendering("bubbleRendering"))
bubbleRendering(args){}

Property: *bubbleRendering*

@Html.EJS().Maps("container").BubbleRendering("bubbleRendering").Render()
bubbleRendering(args){}

| Shape Rendering | **Property:** *shapeRendering*

@(Html.EJ().Map("container").ShapeRendering("shapeRendering"))
shapeRendering(args){}

Property: *shapeRendering*

@Html.EJS().Maps("container").ShapeRendering("shapeRendering").Render()
shapeRendering(args){}

| Zoomed In | **Property:** *zoomedIn*

@(Html.EJ().Map("container").ZoomedIn("zoomedIn"))
zoomedIn(args){}

| Render Completed | **Property:** *onRenderComplete*

@(Html.EJ().Map("container").OnRenderComplete("onRenderComplete"))
onRenderComplete(args){}

Property: *loaded*

@Html.EJS().Maps("container").Click("click").Render()
click(args){}

| Panned | **Property:** *panned*

 @(Html.EJ().Map("container").Panned("panned"))
function panned(args){}| Not Applicable |

| zoomed Out | **Property:**
zoomedOut

@(Html.EJ().Map("container").ZoomedOut("zoomedOut"))
function
 zoomedOut(args){}| Not Applicable |

| Mouse Over | **Property:** *mouseover*

 @(Html.EJ().Map("container").Mouseover("mouseover"))
function mouseover(args){}| Not
 Applicable |

| Mouse Leave | **Property:**
mouseleave

@(Html.EJ().Map("container").MouseLeave("mouseLeave"))
function
 mouseLeave(args){}| Not Applicable |

| Click | **Property:** *click*

 @(Html.EJ().Map("container").Click("click"))
function
 markerSelected(args){}|

| Double Click | **Property:** *doubleClick*

@(Html.EJ().Map("container").
 DoubleClick("doubleClick"))
function doubleClick(args){}| **Property:** *doubleClick*

 @Html.EJS().Maps("container").DoubleClick("doubleClick").Render()
function
 doubleClick(args){}|

| Right Click | **Property:** *rightClick*

 @(Html.EJ().Map("container").RightClick("rightClick"))
function rightClick(args){}| **Property:**
rightClick

 @Html.EJS().Maps("container").RightClick("rightClick").Render()
function rightClick(args){}|

| Initial Load | **Property:** *onLoad*

 @(Html.EJ().Map("container").
 OnLoad("onLoad"))
function onLoad(args){}| **Property:** *load*

 @Html.EJS().Maps("container").Load("mapLoad").Render()
function mapLoad(args){}|

| Before Print | Not Applicable | **Property:** *beforePrint*

 @Html.EJS().Maps("container").BeforePrint("beforePrint").Render()
function
 beforePrint(args){}|

| Resize | Not Applicable | **Property:** *resize*

 @Html.EJS().Maps("container").Resize("resize").Render()
function resize(args){}|

| Tooltip Render | Not Applicable | **Property:**
tooltipRender

@Html.EJS().Maps("container").TooltipRender("tooltipRender").Render()

function tooltipRender(args){}|

| Item Selection | Not Applicable | **Property:** *itemSelection*

 @Html.EJS().Maps("container").ItemSelection("itemSelection").Render()
function
 itemSelection(args){}|

| Item Highlight | Not Applicable | **Property:** *itemHighlight*

 @Html.EJS().Maps("container").ItemHighlight("itemHighlight").Render()
function
 itemHighlight(args){}|

| Shape Highlight | Not Applicable | **Property:** *shapeHighlight*


```
@Html.EJS().Maps("container").ShapeHighlight("shapeHighlight").Render()<br/>function
shapeHighlight(args){}
```

| Layer Rendering | Not Applicable | **Property:** *layerRendering*


```
@Html.EJS().Maps("container").LayerRendering("layerRendering").Render()<br/>function
layerRendering(args){}
```

| Marker Rendering | Not Applicable | **Property:** *markerRendering*


```
@Html.EJS().Maps("container").MarkerRendering("markerRendering").Render()<br/>function
markerRendering(args){}
```

| Bubble Mouse Move | Not Applicable | **Property:** *bubbleMouseMove*


```
@Html.EJS().Maps("container").BubbleMouseMove("bubbleMouseMove").Render()<br/>function
bubbleMouseMove(args){}
```

| Bubble Mouse Move | Not Applicable | **Property:** *annotationRendering*


```
@Html.EJS().Maps("container").AnnotationRendering("annotationRendering").Render()<br/>function
annotationRendering(args){}
```

| Animation Complete | Not Applicable | **Property:** *animationComplete*


```
@Html.EJS().Maps("container").AnimationComplete("animationComplete").Render()<br/>function
animationComplete(args){}
```

How To

Annotations in ASP.NET MVC Maps Component

Annotations are used to mark the specific area of interest in the Maps with texts, shapes, or images. Any number of annotations can be added to the Maps component.

Initialize the Maps component with annotation option, text content or ID of an HTML element or an HTML string can be specified to render a new element that needs to be displayed in the Maps by using the **Content** property. To specify the content position with **X** and **Y** properties as mentioned in the following example.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@Html.EJS().Maps("maps").Load("mapsLoad").Layers(layer =>
{
    layer.ShapeSettings(s =>
    s.Fill("url(#grad1)").ShapeData(ViewBag.africa).Add();
    }).Render()
<script>
    function mapsLoad(args) {
        var maps = args.maps;
        args.maps.annotations = [
            {
                content: '#maps-annotation',
                x: '0%', y: '70%'
            },
            {
                content: '#compass-maps',
                x: '80%', y: '5%'
            }
        ]
    }
}
```



```

    ];
}
}
</script>
<svg height="150" width="400">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="0%" y2="100%">
      <stop offset="0%" style="stop-color:#C5494B;stop-
opacity:1"></stop>
      <stop offset="100%" style="stop-color:#4C134F;stop-
opacity:1"></stop>
    </linearGradient>
  </defs>
</svg>
<div id="maps-annotation" style="display: none;">
  <div id="annotation">
    <div>
      <p style="margin-left:10px;font-size:13px;font-weight:500">Facts
about Africa</p>
    </div>
    <hr style="margin-top:-3px;margin-bottom:10px;border:0.5px solid
#DDDDDD">
    <div>
      <ul style="list-style-type:disc; margin-left:-20px;margin-
bottom:2px; font-weight:400">
        <li>Africa is the second largest and second most populated
continent in the world.</li>
        <li style="padding-top:5px;">
          Africa has 54 sovereign states and 10 non-sovereign
territories.
        </li>
        <li style="padding-top:5px;">Algeria is the largest country
in Africa, where as Mayotte is the smallest.</li>
      </ul>
    </div>
  </div>
</div>
<div id="compass-maps" style="display: none;">
  
</div>
<style>
  #annotation {
    color: #DDDDDD;
    font-size: 12px;
    font-family: Roboto;
    background: #3E464C;
    margin: 20px;
    padding: 10px;
    border-radius: 2px;
    width: 300px;
    box-shadow: 0px 2px 5px #666;
  }
</style>

```

HOWTO-ANNOTATION.CS

```

using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.africa = GetAfricaMap();
            return View();
        }
        public object GetAfricaMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Africa.json");
            return JsonConvert.DeserializeObject(text);
        }
    }
}

```



Custom path map in ASP.NET MVC Maps Component

Maps component can be customized as the desired layout using the custom path map feature. Here, the Maps component has been showcased with normal geometry type shapes to represent the bus seat selection layout.

CSHTML

```

@using Syncfusion.EJ2.Maps
@using Syncfusion.EJ2
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        GeometryType = GeometryType.Normal,

```

```

        ShapeData = ViewBag.seatData,
        ShapeSettings = new MapsShapeSettings
        {
            ColorValuePath = "fill"
        },
    } }).Render()
<div class="col-lg-9">
    <div style="width:200px;margin:auto;padding-bottom:20px">
        
        <div style="padding-left:30px;font-size:20px;font-weight:400;">Bus
seat selection</div>
    </div>
    <div style="border: 3px solid
darkgray;width:200px;display:block;margin:auto;border-radius:5px">
        
    </div>
</div>

```

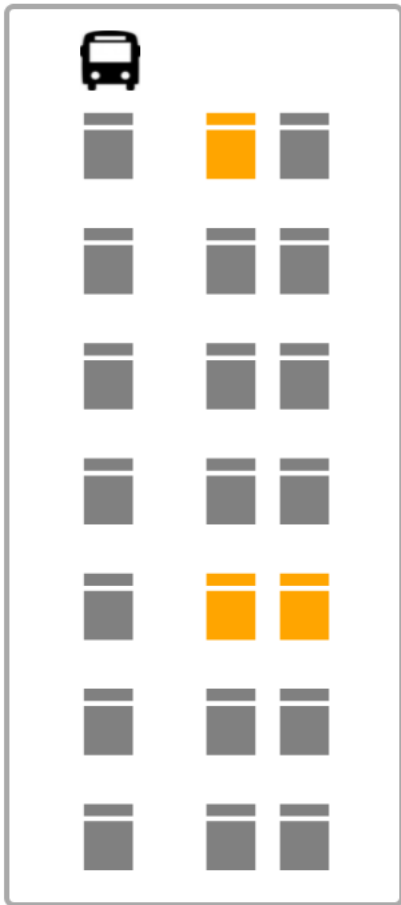
CUSTOM-PATH.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.seatData = GetSeatData();
            return View();
        }
        public object GetSeatData()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/seat.json");
            return JsonConvert.DeserializeObject(text);
        }
    }
}

```

Bus seat selection



Drill-down in ASP.NET MVC Maps Component

By clicking a continent, all the countries available in that continent can be viewed using the drill-down feature. For example, the countries in the **Africa** continent have been showcased here. To showcase all the countries in **Africa** continent by clicking the **ShapeSelected** event as mentioned in the following example.

CSHTML

```
@{ var data = new[]
{
    new { drillColor = "#C13664", continent = "North America",
    CategoryName = "Books", Sales = 10882,
    color="#71B081", visibility = true},
    new { drillColor = "#9C3367", continent = "South America",
    CategoryName = "Books", Sales = 13776,
    color="#5A9A77", visibility = true},
    new { drillColor = "#80306A", continent = "Europe", CategoryName =
    "Books", Sales = 3746,
    color="#39776C", visibility = false},
```

```

        new { drillColor = "#462A6D", continent = "Asia", CategoryName =
"Books", Sales = 10688,
            color="#266665", visibility = false},
        new { drillColor = "#80306A", continent = "Africa", CategoryName =
"Books", Sales = 18718,
            color="#498770", visibility = true},
        new { drillColor = "#2A2870", continent = "Australia", CategoryName
= "Books", Sales = 30716,
            color="#124F5E", visibility = false}
    };
    var markerData = new[]
    {
        new {latitude = 10.97274101999902, longitude = 16.390625}
    };
}
@Html.EJS().Maps("maps").ShapeSelected("shapeSelected").Layers(l =>
{
    l.ShapeSettings(ss => ss.ColorValuePath("drillColor")).MarkerSettings(m
=>m.Visible(true).Template("<div id='marker3'
class='markerTemplate'>Africa</div>").DataSource(markerData).AnimationDurati
on(0).Add()).ShapeDataPath("continent").ShapePropertyPath("continent").DataS
ource(data).ShapeData(ViewBag.worldMap).Add();
    l.ShapeSettings(ss =>
ss.Fill("#80306A")).ShapeData(ViewBag.africaMap).Add();
}).Render()
<script>
    function shapeSelected(args) {
        window.maps = args.maps;
        let shape = args.shapeData.continent;
        if (shape === 'Africa') {
            window.maps.baseLayerIndex = 1;
            window.maps.refresh();
        }
    }
</script>
<style>
    .markerTemplate {
        font-size: 12px;
        color: white;
        text-shadow: 0px 1px 1px black;
        font-weight: 500
    }
    .markerTemplate {
        height: 30px;
        width: 30px;
        display: block;
        margin: auto;
    }
</style>

```

DRILLDOWN.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;

```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.worldmap = GetWorldMap();
            ViewBag.africa = GetAfricaMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetAfricaMap()
        {
            string text =
System.IO.File.ReadAllText("./wwwroot/scripts/MapsData/Africa.json");
            return JsonConvert.DeserializeObject(text);
        }
    }
}
```



Marker types in ASP.NET MVC Maps Component

Add different types of markers

Different marker objects can be added to the Maps component using the marker settings. To update different marker settings in Maps, follow the given steps.

Step 1:

Initialize the Maps component with marker settings. Here, a marker has been added with specified latitude and longitude of California by using the `DataSource` property. To customize the shape of the marker using the `Shape` property and change the border color and width of the marker using the `Border` property as mentioned in the following example.

CSHTML

```
@{
    var data = new[]
    {
        new { latitude= 40.7424509, longitude = -74.0081468, city = "New
York"}
    };
    var border = new Syncfusion.EJ2.Maps.MapsBorder
    {
        Width = 2,
        Color = "#333",
        Opacity = 1
    };
}
@using Syncfusion.EJ2
```

```

@using Syncfusion.EJ2.Maps
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.usMap,
        MarkerSettings = new List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data,
                Width= 2,
                AnimationDuration= 0,
                Border = border,
                Fill = "white",
                Shape =
Syncfusion.EJ2.Maps.MarkerType.Circle
            }
        }
    }
}).Render()

```

MARKER1.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usmap = GetUSMap();
            ViewBag.usMap = GetMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```


**Step 2:**

Customize the above option for n number of markers as mentioned in the following example.

C#HTML

```
@{
    var data = new[]
    {
        new { latitude= 40.7424509, longitude = -74.0081468, city = "New
York"}
    };
    var data1 = new[]
    {
        new { latitude= 33.5302186, longitude = -117.7418381, city =
"Laguna Niguel"}
    };
    var data2 = new[]
    {
        new { latitude= 37.6276571, longitude = -122.4276688, city = "San
Bruno"}
    };
    var border = new Syncfusion.EJ2.Maps.MapsBorder
    {
        Width = 2,
        Color = "#333",
        Opacity = 1
    };
    var border1 = new Syncfusion.EJ2.Maps.MapsBorder
```

```

    {
        Width = 2,
        Color = "#333",
        Opacity = 1
    };
    var border2 = new Syncfusion.EJ2.Maps.MapsBorder
    {
        Width = 2,
        Color = "blue",
        Opacity = 1
    };
}
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Maps
@Html.EJS().Maps("container").Layers(new List<Syncfusion.EJ2.Maps.MapsLayer>
{
    new Syncfusion.EJ2.Maps.MapsLayer
    {
        ShapeData = ViewBag.usMap,
        MarkerSettings = new List<Syncfusion.EJ2.Maps.MapsMarker>
        {
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data2,
                Width= 3,
                AnimationDuration= 0,
                Border = border,
                Fill = "white",
                Shape =
Syncfusion.EJ2.Maps.MarkerType.Circle
            },
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data1,
                Width= 15,
                Height = 4,
                AnimationDuration= 0,
                Border = border,
                Fill = "yellow",
                Shape =
Syncfusion.EJ2.Maps.MarkerType.Rectangle
            },
            new Syncfusion.EJ2.Maps.MapsMarker
            {
                Visible = true,
                DataSource = data,
                Width= 10,
                Height = 10,
                AnimationDuration= 0,
                Border = border,
                Fill = "white",
                Shape =
Syncfusion.EJ2.Maps.MarkerType.Diamond
            }
        }
    })
}).Render()

```

MARKER2.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usmap = GetUSMap();
            ViewBag.usMap = GetMap();
            return View();
        }
        public object GetUSMap()
        {
            string allText =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.json");
            return JsonConvert.DeserializeObject(allText);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/USA.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}
```



Adding multiple layers in the Map

The multilayer support allows loading multiple shape files in a single container and enables Maps to display more information. The shape layer is the main layer of the Maps. Multiple layers can be added in a shape layer as **SubLayer** using the **Type** property.

CSHTML

```
@using Syncfusion.EJ2.Maps;
@using Syncfusion.EJ2;
@{
    var shapeSettings = new Syncfusion.EJ2.Maps.MapsShapeSettings
    {
        Fill = "#E5E5E5",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "black",
            Width = 0.1,
            Opacity = 1
        }
    };
    var shapeSettings1 = new Syncfusion.EJ2.Maps.MapsShapeSettings
    {
        Fill = "rgba(141, 206, 255, 0.6)",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "#1a9cff",
            Width = 0.25,
            Opacity = 1
        }
    };
}
```

```

    }
    };
    var shapeSettings2 = new Syncfusion.EJ2.Maps.MapsShapeSettings
    {
        Fill = "rgba(141, 206, 255, 0.6)",
        Border = new Syncfusion.EJ2.Maps.MapsBorder
        {
            Color = "#1a9cff",
            Width = 0.25,
            Opacity = 1
        }
    };
}
@Html.EJS().Maps("maps").Layers(l =>
{
    l.ShapeSettings(shapeSettings).ShapeData(ViewBag.usmap).Add();

    l.ShapeSettings(shapeSettings1).Type(Syncfusion.EJ2.Maps.Type.SubLayer).ShapeData(ViewBag.california).Add();

    l.ShapeSettings(shapeSettings2).Type(Syncfusion.EJ2.Maps.Type.SubLayer).ShapeData(ViewBag.texas).Add();
}).Render()

```

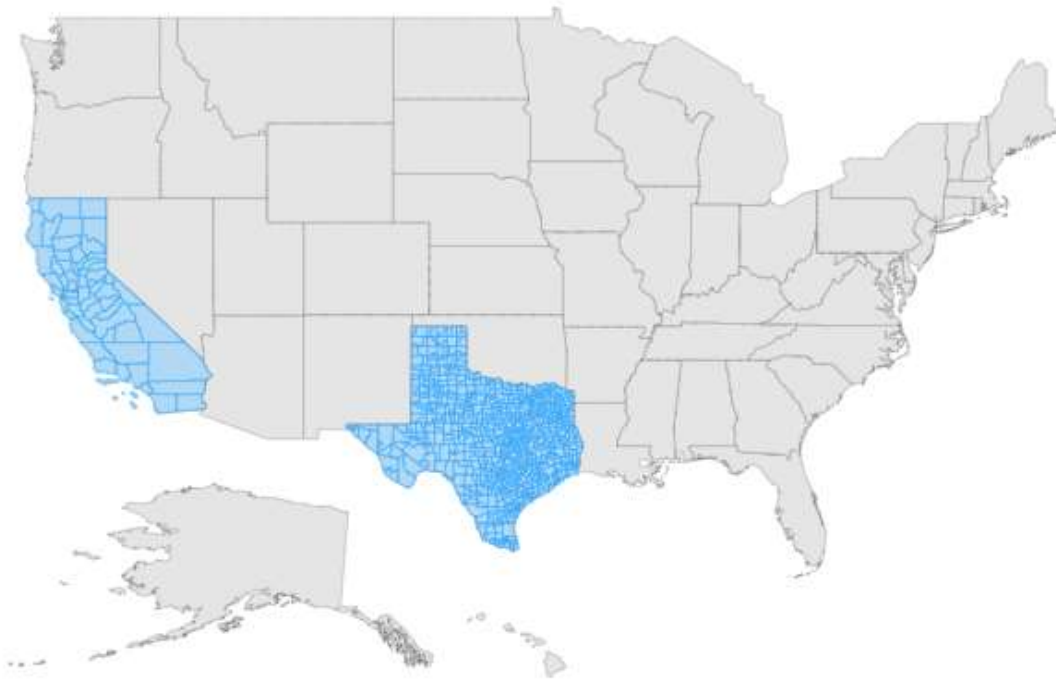
MULTILAYER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.usa = GetUSMap();
            ViewBag.california = GetCaliforniaMap();
            ViewBag.texas = GetTexasMap();
            return View();
        }
        public object GetUSMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/USA.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetCaliforniaMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/California.json");

```

```
        return JsonConvert.DeserializeObject(text);
    }
    public object GetTexasMap()
    {
        string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/Texas.json");
        return JsonConvert.DeserializeObject(text);
    }
}
```



Center position zooming

The center position zooming can be achieved by using the `MapsCenterPosition` class and `ZoomFactor` property as mentioned in the following example. The center position is used to configure the zoom level of Maps, and the zoom factor is used to specify the center position where the Maps should be displayed.

CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Maps;
@{
    var zoom = new Syncfusion.EJ2.Maps.MapsZoomSettings
    {
        Enable = true,
        ZoomFactor = 13
    };
    var centerPosition = new Syncfusion.EJ2.Maps.MapsCenterPosition
```

```

        {
            Latitude = 25.5424414701248,
            Longitude = -89.62646484375
        };
    }
    @Html.EJS().Maps("maps").ZoomSettings(zoom).CenterPosition(centerPosition).Layers(l =>
    {
        l.ShapeData(ViewBag.worldMap).Add();
    }).Render()

```

CENTERPOSITION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
using Syncfusion.EJ2.Charts;
namespace EJ2_Core_Application.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
            ViewBag.world_map = GetWorldMap();
            ViewBag.worldMap = GetMap();
            return View();
        }
        public object GetWorldMap()
        {
            string text =
System.IO.File.ReadAllText("../wwwroot/scripts/MapsData/WorldMap.json");
            return JsonConvert.DeserializeObject(text);
        }
        public object GetMap()
        {
            string allText =
System.IO.File.ReadAllText(Server.MapPath("~/App_Data/WorldMap.json"));
            return JsonConvert.DeserializeObject(allText, typeof(object));
        }
    }
}

```



MaskedTextBox

Getting Started with ASP.NET MVC MaskedTextBox control

This section briefly explains about how to include [ASP.NET MVC MaskedTextBox](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,


```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
,
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

~/_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```


Add ASP.NET MVC MaskedTextBox control

Now, add the Syncfusion ASP.NET MVC MaskedTextBox control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().MaskedTextBox("mask1").Placeholder("MaskedTextBox").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC MaskedTextBox control will be rendered in the default web browser.



Set the mask

You can set the mask to the MaskedTextBox to validate the user input by using the [Mask](#) property. To know more about the usage of mask and configuration, refer to this [link](#).

The following example demonstrates the usage of mask element **0** that allows any single digit from **0** to **9**.

CSHTML

```
@Html.EJS().MaskedTextBox("mask1").Mask("000-000-0000").Placeholder("MaskedTextBox").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
```

MaskedTextBox

Note: [View Sample in GitHub](#).

See also

- [How to perform custom validation using FormValidator](#)
- [How to customize the UI appearance of the control](#)
- [How to set cursor position while focus on the input textbox](#)
- [How to display numeric keypad when focus on mobile devices](#)

Mask Configuration in MaskedTextBox Control

The mask is a combination of standard and custom mask elements that validates the user input based on its behavior.

Note: When the mask value is empty, the MaskedTextBox behaves as an input element with text type.

Standard mask elements

The following table shows the list of mask elements and its behavior based on [MSDN](#) standard.

The mask can be formed by combining any one or more of these mask elements.

| Mask Element | Description |
|--------------|---|
| ----- | ----- |
| 0 | Digit required. This element will accept any single digit from 0 to 9 . |
| 9 | Digit or space, optional. |
| # | Digit or space, optional, Plus(+) and minus(-) signs are allowed. |
| L | Letter required. It will accept letters a-z and A-Z . |
| ? | Letter or space, optional. |
| & | Requires a character. |
| C | Character or space, optional. |
| A | Alphanumeric (A-Za-z0-9) required. |

| a | Alphabetic (**A-Za-z0-9**) or space, optional. |

| < | Shift down. Converts all characters to lower case. |

| > | Shift up. Converts all characters to upper case. |

| | | Disable a previous shift up or shift down. |

| \\\ | Escapes a mask character, turning it into a literal. |

| All other characters | Literals. All non-mask elements (literals) will appear as themselves within MaskedTextBox. |

The following example demonstrates the usage of standard mask elements.

CSHTML

```
@Html.EJS().MaskedTextBox("mask1").Mask("#####").Placeholder("Mask #####
(ex: 012+-
)") .FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
@Html.EJS().MaskedTextBox("mask2").Mask("LLLLLL").Placeholder("Mask LLLLLL
(ex:
Sample)").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render
()
@Html.EJS().MaskedTextBox("mask3").Mask("&&&&&").Placeholder("Mask &&&&&
(ex:
A12@#)").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render(
)
@Html.EJS().MaskedTextBox("mask4").Mask(">LLL<LLL").Placeholder("Mask
>LLL<LL (ex:
SAmple)").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render
()
@Html.EJS().MaskedTextBox("mask5").Mask("\\A999").Placeholder("Mask \\A999
(ex:
A321)").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
```

STANDARDMASKS.CS

```
public ActionResult standardMasks()
{
    return View();
}
```

Output be like the below.

Mask ##### (ex: 012+-)

Mask LLLLLL (ex: Sample)

Mask &&&&& (ex: A12#)

Mask >LLL<LL (ex: SAMple)

Mask \A999 (ex: A321)

A__

Custom mask elements

Other than the above standard mask elements, the mask can be configured with the custom characters or regular expression to define a custom behavior.

Custom characters

You can define any of the non-mask element as the mask element and its behavior through the [customCharacters](#) property.

In the following example, non-mask element **P** accepts the values **P, A, p, a**, and **M** accepts the values **M, m** as mentioned in the custom characters collection.

CSHTML

```
@Html.EJS().MaskedTextBox("mask1").Mask("00:00
>PM").CustomCharacters(ViewBag.cusObj).Placeholder("Time (ex: 10:00 PM,
10:00
AM) ").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
```

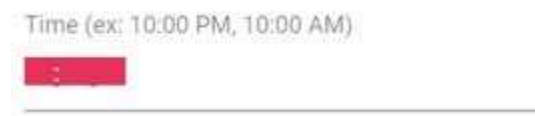
CUSTOM-MASK.CS

```
namespace EJ2CoreSampleBrowser.Controllers.MaskedTextBox
{
    public partial class MaskedTextBoxController : Controller
    {
        public IActionResult CustomMask()
        {
            CustomCharacters customObj = new CustomCharacters();
            customObj.P = "P,A,a,p";
            customObj.M = "M,m";
            ViewBag.cusObj = customObj;
            return View();
        }
    }
}

public class CustomCharacters
```

```
{
    public string P { get; set; }
    public string M { get; set; }
}
```

Output be like the below.



Regular expression

Instead of the mask element, you can define your own regular expression to validate the input of a particular input place. The regular expressions should be wrapped by the square brackets (e.g., [Regex]).

In the following example, regular expression has been set for each input places.

CSHTML

```
<div class="control-section">
    <div class="control-label">IP Address (ex: 212.212.111.222)</div>
    @Html.EJS().MaskedTextBox("mask1").Mask("[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9].[0-2][0-9][0-9]").Render()
</div>
<style>

    .control-label {
        padding-right: 200px;
        font-size: 12px;
    }
    .control-section {
        min-height: 200px;
    }
</style>
```

REGULAREXPRESSION.CS

```
namespace EJ2CoreSampleBrowser.Controllers.MaskedTextBox
{
    public partial class MaskedTextBoxController : Controller
    {
        public IActionResult RegularExpression()
        {
            return View();
        }
    }
}
```

Output be like the below.

IP Address (ex: 212.212.111.222)

234. . .

Prompt character

The Prompt character is a prompting symbol in the MaskedTextBox for the mask elements. The symbol is used to show the input positions in the MaskedTextBox. You can customize the prompt character of MaskedTextBox by using the [promptChar](#) property.

The following example demonstrates the MaskedTextBox with customized prompt character as #.

CSHTML

```
@Html.EJS().MaskedTextBox("mask1").Mask("999-999-9999").PromptChar("#").Render()
```

PROMPT.CS

```
public ActionResult prompt()
{
    return View();
}
```

Output be like the below.

###-###-####

Accessibility

The Maskedtextbox component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Maskedtextbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

```

| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The MaskedTextBox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This control is designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The MaskedTextBox uses the `textbox` role and following ARIA properties for its element based on its state.

| Property | Functionality |

| Property | Functionality |
|------------------|--|
| aria-live | The <code>aria-live</code> attribute indicates the priority of updates to a live region. |
| aria-disabled | The <code>aria-disabled</code> property indicates the disabled state of the MaskedTextBox. |
| aria-valuenow | The <code>aria-valuenow</code> property specifies the current value of the MaskedTextBox. |
| aria-invalid | The <code>aria-invalid</code> property indicates that the user input is incorrect or not within the acceptable ranges. |
| aria-placeholder | The <code>aria-placeholder</code> is a short hint to help the users with data entry when the MaskedTextBox has no value. |
| aria-labelledby | The <code>aria-labelledby</code> property indicates the floating label element of the MaskedTextBox. |

Ensuring accessibility

The MaskedTextBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the MaskedTextBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the MaskedTextBox component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Style and appearance in MaskedTextBox Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of MaskedTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height, font size, and border /
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {
font-size: 20px;
border-color: red;
height: 40px;
border: 2px solid;
}
```

Customizing the MaskedTextBox element on hovering

Use the following CSS to customize the Input Mask element on hovering

```
`css
/ To specify border /
.e-input-group input.e-input, .e-input-group input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):not(:focus), .e-input-group.e-control-wrapper input.e-input, .e-input-group.e-control-wrapper input.e-input:hover:not(.e-success):not(.e-warning):not(.e-error):not([disabled]):not(:focus){
border: 3px solid red;
}
```


How To

Perform custom validation using FormValidator

To perform custom validation on the MaskedTextBox use the FormValidator along with custom validation rules.

In the following example, the MaskedTextBox is validated for invalid mobile number by adding custom validation in the rules collection of the FormValidator.

CSHTML

```
<form id="form-element">
    @Html.EJS().MaskedTextBox("mask1").Mask("000-000-0000").Placeholder("Mobile Number").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Created("onCreate").Render()
    @Html.EJS().Button("submit_btn").Content("Button").Render()
</form>
<script>
    var customFn = function (args) {
        var argsLength = args.element.ej2_instances[0].value.length;
        if (argsLength != 0) {
            return argsLength >= 10;
        }
        else return true;
    };
    var custom = function (args) {
        var argsLength = args.element.ej2_instances[0].value.length;
        if (argsLength == 0) {
            return 0;
        }
        else {
            return argsLength;
        }
    };
    // sets required property in the FormValidator rules collection
    var options = {
        rules: {
            'mask': { numberValue: [customFn, 'Enter valid mobile number']
        },
    },
    };
    // defines FormValidator to validate the MaskedTextBox
    var formObject = new ej.inputs.FormValidator('#form-element', options);
    formObject.addRules('mask', { maxLength: [custom, 'Enter mobile number']
});
    // places error label outside the MaskedTextBox using the customPlacement event of FormValidator
    formObject.customPlacement = function (element, errorElement) {
        document.querySelector("#mask1").parentNode.appendChild(errorElement);
    };
    document.getElementById("submit_btn").addEventListener('click', function () {
        formObject.validate("mask");
        var ele = document.getElementById('mask1');
        // checks for incomplete value and alerts the formt submit
```

```

        if (ele.value !== "" &&
ele.value.indexOf(ele.ej2_instances[0].promptChar) === -1) {
            alert("Submitted");
        }
    });
    function onCreate() {
        document.getElementById(this.element.id).setAttribute("name",
"mask");
    }
</script>

```

HOWTO.CS

```

public ActionResult howto()
{
    return View();
}

```

Output be like the below.



Set cursor position while focus on the input textbox

By default, on focusing the MaskedTextBox the entire mask gets selected. You can customize by using any one of the following methods:

- Setting cursor position at the start of the MaskedTextBox.
- Setting cursor position at the end of the MaskedTextBox.
- Setting cursor at the specified position in the MaskedTextBox.

Note: The **selectionStart** and **selectionEnd** set to **0** instead of the input element value's length, when we focus on a MaskedTextBox control filled with all mask characters. This is the default behavior of the HTML 5 input element.

Following is an example that demonstrates the above cases to set cursor position in the MaskedTextBox using [focus](#) event.

CSHTML

```

@Html.EJS().MaskedTextBox("mask1").Mask("00000-00000").Value("93828-32132").Placeholder("Default cursor position").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
@Html.EJS().MaskedTextBox("mask2").Mask("00000-00000").Value("83929-4342").Placeholder("Cursor positioned at start").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Focus("onfocus").Render()

```

```
@Html.EJS().MaskedTextBox("mask3").Mask("00000-00000").Value("83929-3213").Placeholder("Cursor positioned at end").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Focus("onfocus1").Render()
@Html.EJS().MaskedTextBox("mask4").Mask("+1 000-000-0000").Value("234-432-432").Placeholder("Cursor at specified position").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Focus("onfocus2").Render()
<script>
    function onfocus(args) {
        //sets cursor position at start of MaskedTextBox
        args.selectionEnd = args.selectionStart = 0;
    }
    function onfocus1(args) {
        //sets cursor position at end of MaskedTextBox
        args.selectionStart = args.selectionEnd = args.maskedValue.length;
    }
    function onfocus2(args) {
        //sets cursor at specified position
        args.selectionStart = 3;
        args.selectionEnd = 3;
    }
</script>
```

CURSORPOSITION.CS

```
public ActionResult cursorPosition()
{
    return View();
}
```

Output be like the below.



Display numeric keypad when focus on mobile devices

By default, on focusing the MaskedTextBox, alphanumeric keypad will be displayed on mobile devices. Sometimes only numeric keypad for number values is needed, and this can be achieved by setting "type" property to tel.

Refer to the following example to enable numeric keypad in MaskedTextBox.

CSHTML

```
@Html.EJS().MaskedTextBox("mask1").Mask("000-000-0000").Created("onCreate").Render()
<script>
    function onCreate() {
        var key = document.getElementById("mask1");
        key.type = "tel";
    }
</script>
```

NUMERICKEYPAD.CS

```
public ActionResult numericKeypad()
{
    return View();
}
```

Customize the UI appearance of the control

The appearance of the MaskedTextBox can be changed by adding custom `cssClass` to the control and enabling styles.

Refer to the following example to change the appearance of the MaskedTextBox.

CSHTML

```
@Html.EJS().MaskedTextBox("mask4").Mask("00000").Value("42648").CssClass("e-style").Placeholder("Enter user ID").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Focus("onfocus").Render()
<script>
    function onfocus(args) {
        //sets cursor position at start of MaskedTextBox
        args.selectionEnd = args.selectionStart;
    }
</script>
<style>
    .e-mask.e-style .e-control.e-maskedtextbox {
        color: #00ffff;
        letter-spacing: 10px;
        font-size: xx-large;
        border: 1px;
        border-color: #ffffff;
    }
    .e-control-wrapper.e-mask.e-float-input.e-style .e-float-line::before {
        background: #ffffff;
    }
    .e-control-wrapper.e-mask.e-float-input.e-style .e-float-line::after {
        background: #ffffff;
    }
    .e-control-wrapper.e-mask.e-float-input.e-style .e-float-text.e-label-top {
        color: #00ffff;
        font-size: medium;
    }
```

```
}
</style>
```

CUSTOMCSS.CS

```
public ActionResult customCss()
{
    return View();
}
```

Output be like the below.



MaskedTextBoxFor and Model Binding

This section demonstrates the Strongly typed extension support in MaskedTextBox. The view which bind with any model is called as strongly typed view. You can bind any class as model to view. You can access model properties on that view. You can use data associated with model to render controls.

In this sample, first click the submit button to post the selected value in the MaskedTextBox. When posting the null value, validation error message will be shown below the MaskedTextBox.

CSHTML

```
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)
    <div class="col-lg-12 control-section">
        <div id="wrapper">
            @Html.EJS().MaskedTextBoxFor(model =>
model.value).Width("200px").Created("onCreate").Render()
            <div>
                @Html.ValidationMessageFor(model => model.value)
            </div>
            <div id="submitbutton">
                @Html.EJS().Button("btn").Content("Post").Render()
            </div>
        </div>
    </div>
}
<script>
    function onCreate() {
        document.getElementById(this.element.id).setAttribute("name",
"value");
    }
</script>
<style>
    #wrapper {
        max-width: 246px;
        margin: 30px auto;
        padding-top: 50px;
    }
    #submitbutton {
```

```

        margin-top: 20px;
        margin-left: 65px;
    }
    #control-content #wrapper .field-validation-error {
        color: red;
    }
</style>

```

MODELBINDING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using System.ComponentModel.DataAnnotations;
namespace EJ2CoreSampleBrowser.Controllers.MaskedTextBox
{
    public class MaskValue
    {
        [Required]
        public string value { get; set; }
    }
    public partial class MaskedTextBoxController : Controller
    {
        public ActionResult DefaultFunctionalities()
        {
            MaskValue val = new MaskValue();
            val.value = "10";
            return View(val);
        }
        [HttpPost]
        public ActionResult DefaultFunctionalities(MaskValue model)
        {
            MaskValue val = new MaskValue();
            val.value = model.value;
            return View(val);
        }
    }
}

```

Output be like the below.



Migration from Essential JS 1

This article describes the API migration process of MaskEdit component from Essential JS 1 to Essential JS 2.

Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

```
| Adding custom class | Property: CssClass <br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("9999").InputMode(InputMode.Text).CssClass("custom") |  
Property: CssClass <br /><br />@Html.EJS().MaskedTextBox("mask").Mask("9999").CssClass("custom").Render() |
```

```
| Destroy editor | Not Applicable | Method: destroy<br /><br />@Html.EJS().MaskedTextBox("mask").Mask("00-000").Render()<br /><br />Script<br /><br />var mask = document.getElementById('mask').ej2_instances[0]<br />mask.destroy(); |
```

```
| Disable the maskedit control | Method: disable <br /><br />
/>@Html.EJ().MaskEdit("mask").MaskFormat("0000").Value("1234").InputMode(InputMode.Text)<br />
/><br />Script<br /><br />var maskObj = $("#mask").data("ejMaskEdit");<br />maskObj.disable(); | Can
be achieved using<br />@Html.EJS().MaskedTextBox("mask").Mask("00-
000").Render()<br />Script<br />var maskObj =
document.getElementById("mask").ej2_instances[0];<br />maskObj.enabled= false<br /> |
```

```
| Enable the maskedit control | Method: enable<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("0000").Value("1234").InputMode(InputMode.Text)<br /><br /><b>Script</b><br /><br />var maskObj = $("#mask").data("ejMaskEdit");<br />maskObj.enable(); | Can be achieved using</b><br />@Html.EJS().MaskedTextBox("mask").Mask("00-000").Render()<br /><b>Script</b><br />var maskObj = document.getElementById("mask").ej2_instances[0];<br />maskObj.enabled= true<br /> |
```

```
| Control state | Property: enabled<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("00-000").InputMode(InputMode.Text).Enabled(false) | Property: Enabled<br /><br />@Html.EJS().MaskedTextBox("mask").Mask("00-000").Enabled(false).Render() |
```

```
| Persistence | Property: EnablePersistence<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("00-000").InputMode(InputMode.Text).EnablePersistence(true) | Property: EnablePersistence<br /><br />@Html.EJS().MaskedTextBox("mask").Mask("00-000").EnablePersistence(true).Render() |
```

```
| Triggers when editor is focused in | Event: FocusIn<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("0000").InputMode(InputMode.Text).ClientSideEvents(s => s.FocusIn("onFocus"))<br /><br />Script<br /><br />function onFocus(){} | Event: Focus<br /><br />@Html.EJS().MaskedTextBox("mask").Mask("0000").Focus("onFocus")<br />Script<br />function onFocus(){} |
```

```
| Triggers when editor is focused out | Event: FocusOut<br /><br />
/>@Html.EJ().MaskEdit("mask").MaskFormat("00-00").InputMode(InputMode.Text).ClientSideEvents(s
=> s.FocusOut("onFocusOut"))<br /><br />Script<br /><br />function onFocusOut(){} | Event: Blur<br
/><br />
/>@Html.EJS().MaskedTextBox("mask").Mask("0000").InputMode(InputMode.Text).Blur("onBlur")<br />
Script<br />function onBlur(){} |
```

| Sets height | **Property:** *height*
/>@Html.EJ().MaskEdit("mask").MaskFormat("0000").InputMode(InputMode.Text).Height("30px") |
Can be achieved using,
/>@Html.EJS().MaskedTextBox("mask").Mask("9999").CssClass("custom").Render()
/>e-maskedTextBox.custom{height: 30px;}

```
| HTML Attributes | Property: HtmlAttributes<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("0000").InputMode(InputMode.Text).HtmlAttributes(htmlAttr)<br /><br />Script<br /><br />@{<br />Dictionary<string, object> htmlAttr = new Dictionary<string,
```

object>());
htmlAttr.Add("name", "maskedtextbox");
} | **Can be achieved**

```
by<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").setAttribute("name", "textbox");<br/>} |
```

| Specifies Input mode | **Property:** *InputMode*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("0000").InputMode(InputMode.Password) | Can be achieved
```

```
by<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").setAttribute("type", "password");<br/>} |
```

| Triggers on key press | **Event:** *KeyPress*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("000").InputMode(InputMode.Text).ClientSideEvents(s => s.KeyPress("onKeyPress"))<br /><br />Script<br /><br />function onKeyPress(){} | Can be achieved using native
```

```
event<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").addEventListener("keypress", function () {});<br/>} |
```

| Triggers on key up | **Event:** *KeyUp*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("000").InputMode(InputMode.Text).ClientSideEvents(s => s.KeyUp("onKeyUp"))<br /><br />Script<br /><br />function onKeyUp(){} | Can be achieved using native event<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").addEventListener("keyup", function () {});<br/>} |
```

| Triggers on mouse out in maskedit control | **Event:** *MouseOut*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("000").InputMode(InputMode.Text).ClientSideEvents(s => s.MouseOut("onMouseOut"))<br /><br />Script<br /><br />function onMouseOut(){} | Can be achieved using native
```

```
event<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").addEventListener("mouseout", function () {});<br/>} |
```

| Triggers when mouse over in maskedit control | **Event:** *MouseOver*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("00-00").InputMode(InputMode.Text).ClientSideEvents(s => s.MouseOver("onMouseOver"))<br /><br />Script<br /><br />function onMouseOver(){} | Can be achieved using native
```

```
event<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").addEventListener("mouseover", function () {});<br/>} |
```

| Name of maskedit control | **Property:** *name*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("0000").InputMode(InputMode.Text).Name("pin") | Can be achieved
```

```
using<br/>@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br/>Script<br/>function onCreate() {<br/>document.getElementById("mask").setAttribute("name", "TextBox")<br/><br/>}; |
```

| Triggers on keydown | **Event:** *OnKeyDown*
<br

```
/>@Html.EJ().MaskEdit("mask").MaskFormat("9999-
```



```
9999").InputMode(InputMode.Text).ClientSideEvents(s => s.OnKeyDown("keyDown"))<br /><br />
</Script><br /><br />function keyDown(){} | Can be achieved using native event<br />
@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Render()<br />
</Script><br />function onCreate() {<br />document.getElementById("mask").addEventListener("keydown",
function () {});<br />} |
```

```
| Read only | Property: ReadOnly<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("99-
9999").InputMode(InputMode.Text).ReadOnly(true) | Can be achieved using<br />
@Html.EJS().MaskedTextBox("mask").Enabled(false).Mask("0000").Render()<br />
Css<br />.e-mask<br />background-image: none !important;<br />border-bottom-color: rgba(0, 0, 0, 0.42)
!important;<br />} |
```

```
| Right to left | Property: TextAlign<br /><br />
@Html.EJ().MaskEdit("mask").MaskFormat("9999").InputMode(InputMode.Text).TextAlign("Right") |
Property: EnableRtl<br /><br />
@Html.EJS().MaskedTextBox("mask").Mask("9999").EnableRtl(true).Render() |
```

```
| Sets width | Property: width<br /><br />
@Html.EJ().MaskEdit("mask").MaskFormat("0000").InputMode(InputMode.Text).Width("100px") |
Property: Width<br /><br />
@Html.EJS().MaskedTextBox("mask").Mask("0000").Width("100px").Render() |
```

Mask Configuration

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

```
| Triggers on value change | Event Change<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("00-
00").InputMode(InputMode.Text).ClientSideEvents(s => s.Change("onChange"))<br /><br />
</Script><br />function onChange(){} | Event: Change<br /><br />
@Html.EJS().MaskedTextBox("mask").Mask("00-00").Change("onChange").Render()<br /><br />
</Script><br /><br />function onChange(){} |
```

```
| Clears maskedit text/value | Method: clear<br /><br />
@Html.EJ().MaskEdit("mask").MaskFormat("0000").Value("1234").InputMode(InputMode.Text)<br />
<br /></Script><br /><br />var maskObj = $("#mask").data("ejMaskEdit");<br />maskObj.clear(); | Can be achieved using<br />
@Html.EJS().MaskedTextBox("mask").Mask("00-0000").Value("1234").Render()<br />
</Script><br />var maskObj = document.getElementById("mask").ej2_instances[0];<br />maskObj.value = ""<br /> |
```

```
| Triggers on creation | Event: Create<br /><br />@Html.EJ().MaskEdit("mask").MaskFormat("00-
00").InputMode(InputMode.Text).ClientSideEvents(s => s.Create("onCreate"))<br /><br />
</Script><br />function onCreate(){} | Event: Created<br /><br />
@Html.EJS().MaskedTextBox("mask").Mask("00-00").Created("onCreated").Render()<br /><br />
</Script><br /><br />function onCreated(){} |
```

```
| Custom Character | Property: CustomCharacter<br /><br />
@Html.EJ().MaskEdit("mask").MaskFormat("C-0000").InputMode(InputMode.Text).CustomCharacter("#") | Property: CustomCharacters<br /><br />
@Html.EJS().MaskedTextBox("mask").Mask("C-0000").CustomCharacters(ViewBag.cusObj).Render()<br /><br />
</Controller><br /><br />CustomCharacters customObj = new CustomCharacters();<br />customObj.C = "#";<br />
ViewBag.cusObj = customObj; |
```

| Triggers when maskedit control is destroyed | **Event:** *Destroy*
 />@Html.EJ().MaskEdit("mask").MaskFormat("00-00").InputMode(InputMode.Text).ClientSideEvents(s
 => s.Destroy("onDestroy"))

Script

function onDestroy(){ | **Event:** *Destroyed*
 />
@Html.EJS().MaskedTextBox("mask").Mask("00-00").Destroyed("onDestroy").Render()

 />**Script**

function onDestroy(){ |

| Placeholder float type | Not Applicable | **Property:** *floatLabelType*
 />@Html.EJS().MaskedTextBox("mask").Mask("9,999").Placeholder("Enter
 value").FloatLabelType("Auto").Render() |

| Gets pure value of maskedit control | **Method:** *getStrippedValue*
 />@Html.EJ().MaskEdit("mask").Mask("000").Value("1234").InputMode(InputMode.Text)

 />**Script**

var maskObj = \$("#mask").data("ejMaskEdit");
maskObj.getStrippedValue(); |
Can be achieved using
@Html.EJS().MaskedTextBox("mask").Mask("00-
 000").Value("1234").Render()
Script
var maskObj =
 document.getElementById("mask").ej2_instances[0];
alert(maskObj.value)
 |

| Get whole maskedit value | **Method:** *getUnstrippedValue*
 />@Html.EJ().MaskEdit("mask").MaskFormat("00-00").Value("1234").InputMode(InputMode.Text)

 />
Script

var maskObj = \$("#mask").data("ejMaskEdit");

 />maskObj.getUnstrippedValue(); | **Method:** *getMaskedValue*
 />@Html.EJS().MaskedTextBox("mask").Mask("00-000").Render()

Script

var mask
 = document.getElementById('mask').ej2_instances[0]
mask.getMaskedValue(); |

| Hides prompt character on focus out | **Property:** *HidePromptOnLeave*
 />@Html.EJ().MaskEdit("mask").MaskFormat("aaaa").InputMode(InputMode.Text).HidePromptOnLeave
 (true) | **Can be achieved**
using
@Html.EJS().MaskedTextBox("mask").Mask("0000").Created("onCreate").Focus("onFocus").R
 ender()
Script
function onCreate() {
var maskObj =
 this;
document.getElementById("mask").addEventListener("focusout", function ()
 {
maskObj.promptChar = " ";
});

function onFocus() {
this.promptChar =
 " _";
}} |

| Mask format | **Property:** *maskFormat*
 />@Html.EJ().MaskEdit("mask").MaskFormat("99,999") | **Property:** *mask*
 />@Html.EJS().MaskedTextBox("mask").Mask("99,999").Render() |

| Prompt character | Not Applicable | **Property:** *PromptChar*
 />@Html.EJS().MaskedTextBox("mask").Mask("99,999").PromptChar("#").Render() |

| Clear Button | Not Applicable | **Property:** *ShowClearButton*
 />@Html.EJS().MaskedTextBox("mask").Mask("99,999").ShowClearButton(true).Render() |

| Prompt character display | **Property:** *ShowPromptChar*
 />@Html.EJ().MaskEdit("mask").MaskFormat("99-999").ShowPromptChar(false) |
 @Html.EJS().MaskedTextBox("mask").Mask("99999").PromptChar("#").Render() |

| Show rounded corner | **Property:** *ShowRoundedCorner*
 />@Html.EJ().MaskEdit("mask").MaskFormat("0000").ShowRoundedCorner(true) | **Can be achieved**
using
@Html.EJS().MaskedTextBox("mask").Mask("0000").Render()
Css
#mask
 {
border: 2px solid grey;
padding: 7px;
border-radius: 10px;
}
e-control-
 wrapper.e-mask.e-float-input.e-style .e-float-line::before,e-control-wrapper.e-mask.e-float-input.e-
 style .e-float-line::after {
 background: none;
}
 |

| Value of maskedit control | **Property:** *value* @Html.EJ().MaskEdit("mask").MaskFormat("0000").Value("1234") | **Property:** *value* @Html.EJS().MaskedTextBox("mask").Mask("0000").Value("1234").Render() |

| Displays hint on maskedit control | **Property:** *WatermarkText* @Html.EJ().MaskEdit("mask").MaskFormat("9999").WatermarkText("Enter value") | **Property:** *Placeholder* @Html.EJS().MaskedTextBox("mask").Mask("9999").Placeholder("Enter value").Render() |

Validation

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

|-----|-----|-----|

| Displays error until correct value is entered | **Property:** *show-error* @Html.EJ().MaskEdit("mask").MaskFormat("99-999").ShowError(true) | **MaskedTextBox by default shows error until correct value is entered** @Html.EJS().MaskedTextBox("mask").Mask("0000").Render() |

| Validation message | **Property:** *validation-message* @Html.EJ().MaskEdit("mask").MaskFormat("0000").ValidationRules(rule => rule.AddRule("required", true)).ValidationMessage(msg => msg.AddMessage("required", "Required value")) | **Validation in MaskedTextBox can be achieved by Form validation** <form id="form-element"> @Html.EJS().MaskedTextBox("mask").Mask("000-000-0000").Placeholder("Mobile Number").FloatLabelType(FloatLabelType.Always).Created("onCreate").Render() </form> **Script** var options = { rules: { 'mask': { required: [true, 'Enter valid mobile number'] }, }, }; var formObject = new ej.inputs.FormValidator('#form-element', options); formObject.customPlacement = function (element, errorElement) { document.querySelector("#mask").parentNode.appendChild(errorElement); }; function onCreate() { document.getElementById(this.element.id).setAttribute("name", "mask"); } |

| Validation Rules | **Property:** *validation-rules* @Html.EJ().MaskEdit("mask").MaskFormat("0000").ValidationRules(rule => rule.AddRule("required", true)) | **Validation in MaskedTextBox can be achieved by Form validation** <form id="form-element"> @Html.EJS().MaskedTextBox("mask").Mask("000-000-0000").Placeholder("Mobile Number").FloatLabelType(FloatLabelType.Always).Created("onCreate").Render() </form> **Script** var options = { rules: { 'mask': { required: [true] }, }, }; var formObject = new ej.inputs.FormValidator('#form-element', options); formObject.customPlacement = function (element, errorElement) { document.querySelector("#mask").parentNode.appendChild(errorElement); }; function onCreate() { document.getElementById(this.element.id).setAttribute("name", "mask"); } |

Mention

Getting Started with ASP.NET MVC Mention Control

This section briefly explains about how to include [ASP.NET MVC Mention](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC

controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Mention control

Now, add the Syncfusion ASP.NET MVC Mention control in `~/Home/Index.cshtml` page.

CSHTML

```
@{
    char nameMentionChar = '@';
}
<label style="font-size: 15px; font-weight: 600;">Comments</label>
<div id="defaultMention" placeholder="Type @Html.Raw("@") and tag user"
style="min-height: 100px; border: 1px solid #D7D7D7; border-radius: 4px;
padding: 8px; font-size: 14px; width: 600px;"></div>
@Html.EJS().Mention("default").MentionChar(nameMentionChar).Target("#defaultMention").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Mention control will be rendered in the default web browser.

Binding data source

After initialization, populate the Mention with data using the [DataSource](#) property. Here, an array of string values is passed to the Mention control.

CSHTML

```
@model List<string>
```

```
@{
    char nameMentionChar = '@';
}
<label style="font-size: 15px; font-weight: 600;">Comments</label>
<div id="defaultMention" placeholder="Type @Html.Raw("@") and tag user"
style="min-height: 100px; border: 1px solid #D7D7D7; border-radius: 4px;
padding: 8px; font-size: 14px; width: 600px;"></div>
@Html.EJS().Mention("default_data").MentionChar(nameMentionChar).Target("#de
faultMention").DataSource((IEnumerable<object>)ViewBag.data).Render()
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.data = new string[] { "Selma Rose", "Garth", "Robert",
"William", "Joseph" };
            return View();
        }
    }
}
```

Display mention character

By using the [ShowMentionChar](#) property, the text content can be displayed along with the mention character. You can customize the mention character by using the [MentionChar](#) property in the Mention control.

Note: By default, the [MentionChar](#) is @ and the [ShowMentionChar](#) property is disabled.

The following example, displays the text content along with the mention character configured as #.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '#';
}
<label style="font-size: 15px; font-weight: 600;">Comments</label>
<div id="defaultMention" placeholder="Type @Html.Raw("#") and tag user"
style="min-height: 100px; border: 1px solid #D7D7D7; border-radius: 4px;
padding: 8px; font-size: 14px; width: 600px;"></div>
@Html.EJS().Mention("showMention").Target("#defaultMention").MentionChar(nam
eMentionChar).ShowMentionChar(true).DataSource((IEnumerable<object>)ViewBag.
data).Render()
```

SUGGESTIONLIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.data = new string[] { "Selma Rose", "Garth", "Robert",
"William", "Joseph" };
            return View();
        }
    }
}

```

Working with Data in Mention

The Mention loads the data either from local data source or remote data services using the [DataSource](#) property. It supports the data type of either `array` or `DataManager`.

The Mention also supports different kinds of data services such as OData V4 and Web API, and data formats such as XML, JSON, and JSONP with the help of `DataManager` adaptors.

| Fields | Type | Description |
|---------|------------------|--|
| text | string | Specifies the display text of each list item. |
| value | number or string | Specifies the hidden data value mapped to each list item that should contain a unique value. |
| groupBy | string | Specifies the category under which the list item has to be grouped. |
| iconCss | string | Specifies the icon class of each list item. |

Note: When binding complex data to the Mention, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in three ways as described in the following.

Array of simple data

The Mention has provided support to load an array of primitive data such as strings and numbers. Here, both the value and text fields act the same.

CSHTML

```

@model List<string>
@{
    char nameMentionChar = '@';
    string[] data = new string[] { "Selma Rose", "Garth", "Robert",
"William", "Joseph" };
}

```

```

<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("Array-Of-Simple-
Data").MentionChar(nameMentionChar).Target("#mentionElement").DataSource((IE
numerable<object>)data).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>

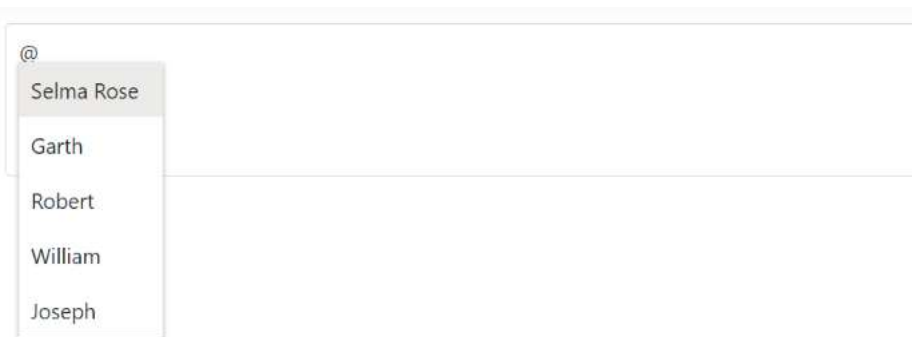
```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        {
            public ActionResult Index()
            {
                return View();
            }
        }
    }
}

```



Array of JSON data

The Mention can generate its list of items through an array of JSON data. Therefore the appropriate columns should be mapped to the [Fields](#) property.

In the following example, **ID** column and **Game** column from complex data have been mapped to the **Value** field and **Text** field, respectively.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
    List<SportsData> data = new SportsData().SportsList();
}

<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag sport"></div>
@Html.EJS().Mention("Array-Of-Json-Data").MentionChar(nameMentionChar).Target("#mentionElement").DataSource((IEnumerable<object>)data).Fields(new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Game" , Value = "ID" }).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

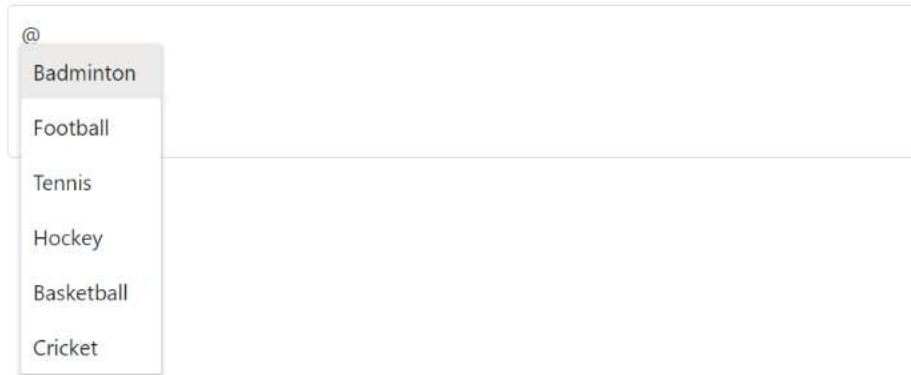
DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class SportsData
    {
        public string ID { get; set; }
        public string Game { get; set; }
        public List<SportsData> SportsList()
        {
            List<SportsData> sports = new List<SportsData>()
            {
                new SportsData { ID = "game1", Game = "Badminton" },
                new SportsData { ID = "game2", Game = "Football" },
                new SportsData { ID = "game3", Game = "Tennis" },
                new SportsData { ID = "game4", Game = "Hockey" },
                new SportsData { ID = "game5", Game = "Basketball" },
            };
            return sports;
        }
    }
}
```

```

    }
  }
}

```



Array of complex data

The Mention can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [Fields](#) property.

In the following example, `Code.ID` and `Country.Name` columns from the complex data have been mapped to the `Value` and `Text` fields respectively.

CSHTML

```

@model List<string>
@{
    char nameMentionChar = '@';
    List<CountryGroup> data = new CountryGroup().GetData();
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
country"></div>
@Html.EJS().Mention("Complex-data-
biding").MentionChar(nameMentionChar).Target("#mentionElement").DataSource((
IEnumerable<object>)data).Fields(new
Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Country.Name", Value
= "Code.Id" }).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>

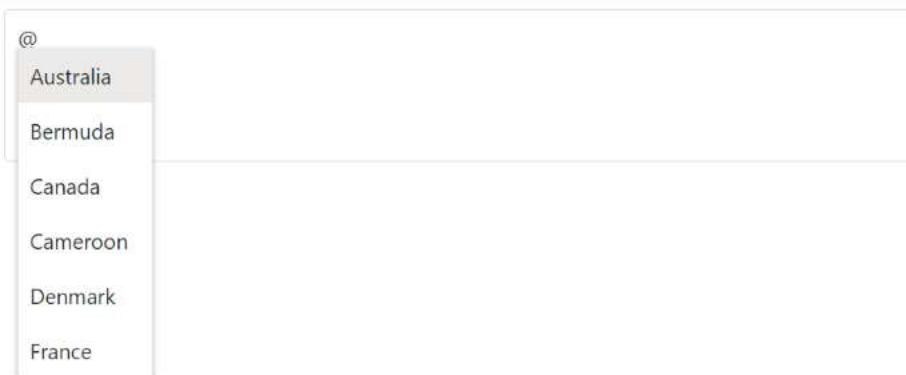
```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Code
    {
        public string ID { get; set; }
    }
    public class Country
    {
        public string Name { get; set; }
    }
    public class CountryGroup
    {
        public Country Country { get; set; }
        public Code Code { get; set; }
        public List<CountryGroup> GetData()
        {
            List<CountryGroup> data = new List<CountryGroup>
            {
                new CountryGroup() { Country = new Country() { Name =
"Australia" }, Code = new Code() { ID = "AU" } },
                new CountryGroup() { Country = new Country() { Name =
"Bermuda" }, Code = new Code() { ID = "BM" } },
                new CountryGroup() { Country = new Country() { Name =
"Canada" }, Code = new Code() { ID = "CA" } },
                new CountryGroup() { Country = new Country() { Name =
"Cameroon" }, Code = new Code() { ID = "CM" } },
                new CountryGroup() { Country = new Country() { Name =
"Denmark" }, Code = new Code() { ID = "DK" } },
                new CountryGroup() { Country = new Country() { Name =
"France" }, Code = new Code() { ID = "FR" } }
            };
            return data;
        }
    }
}

```



Binding remote data

The Mention supports retrieval of data from remote data services with the help of **DataManager** control. The [Query](#) property is used to fetch the data from the database and bind it to the Mention control.

OData v4 adaptor - Binding OData v4 service

The ODataV4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume OData v4 services. For more details on OData v4 services, refer to the [odata documentation](#). To bind OData v4 service, use the **ODataV4Adaptor**.

The following sample displays the first 6 contacts from **Customers** table of the **Northwind** Data Service.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
    var query = "new
ej.data.Query().from('Employees').select('EmployeeID,FirstName,Title').take(
6)";
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").Target("#mentionElement").MentionChar(nameMe
ntionChar).DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor")).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string)query).Render()
<style>
div#mentionElement[placeholder]:empty:before {
    content: attr(placeholder);
}
#mentionElement {
    min-height: 100px;
    border: 1px solid #D7D7D7;
    border-radius: 4px;
    padding: 8px;
    font-size: 14px;
    width: 600px;
}
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
```

```
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

@

Nancy

Andrew

Janet

Margaret

Steven

Michael

Web API adaptor

You can use **WebApiAdaptor** to bind mention with Web API created using OData endpoint.

CSHTML

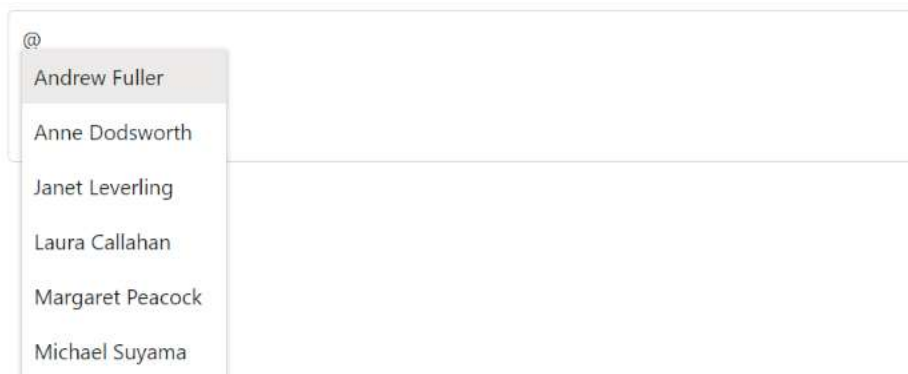
```
@model List<string>
@{
    char nameMentionChar = '@';
    var query = "new ej.data.Query().select(['FirstName','Country',
'EmployeeID']).take(6).requiresCount()";
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").MentionChar(nameMentionChar).Target("#mentio
nElement").DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor").Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string)query).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
```

```
width: 600px;
}
```

</style>

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



See also

- [Customization](#)
- [How to perform filtering](#)

Filtering data in Mention

The Mention control has built-in support to filter data items. The filter operation starts as soon as you start typing characters in the mention element.

Limit the minimum filter character

You can control the minimum length of user input to initiate the search action using the [MinLength](#) property. This can be useful if you have a very large list of data. The default value is 0, where the suggestion list opens as soon as the user inputs the mention character.

The remote request does not fetch the search data until the search key contains three characters as shown in the following example.

CSHTML

```

@model List<string>
@{
    char nameMentionChar = '@';
    var query = "new ej.data.Query().select(['FirstName','Country',
'EmployeeID']).take(6).requiresCount()";
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").MinLength(3).MentionChar(nameMentionChar).Ta
rget("#mentionElement").DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor")).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string)query).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>

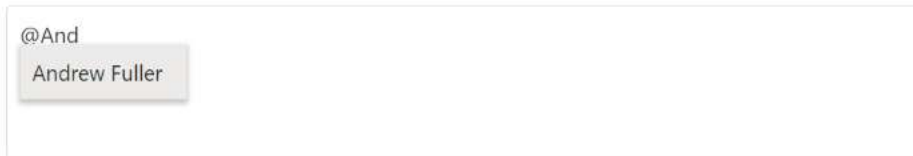
```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```



Change the filter type

While filtering, you can change the filter type to **Contains**, **StartsWith**, or **EndsWith** in the [FilterType](#) property. The default filter operator is **Contains**.

- **StartsWith** - Filter the items that begin with the specified text value.
- **Contains** - Filter the items that contain the specified text value.
- **EndsWith** - Filter the items that end with the specified text value.

CSHTML

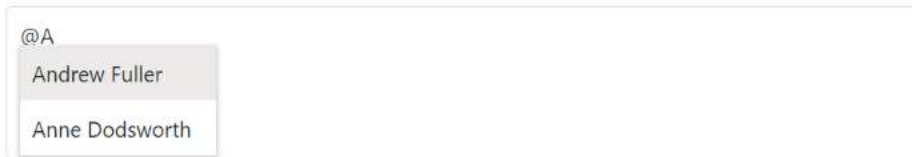
```
@model List<string>
@{
    char nameMentionChar = '@';
    var query = "new ej.data.Query().select(['FirstName', 'Country',
'EmployeeID']).take(6).requiresCount()";
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").FilterType(FilterType.StartsWith).Target("#m
entionElement").MentionChar(nameMentionChar).DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor")).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string)query).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```



```
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



Allow spacing between search

While filtering the data in the data source, you can allow the space in the middle of the mention using the [AllowSpaces](#) property. If the data source does not match with the mentioned element data, the popup will be hidden on the space key press. The default value of the [AllowSpaces](#) is `false`.

Note: By default, the [AllowSpaces](#) property is disabled, and the space ends the mention control search.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
    List<Employees> data = new Employees().GetData();
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag employee"></div>
@Html.EJS().Mention("Allow-
Space").Target("#mentionElement").MentionChar(nameMentionChar).Fields(new
Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Name"
}).DataSource((IEnumerable<object>)data).AllowSpaces(true).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Employees
    {
        public string Name { get; set; }
        public string ID { get; set; }
        public List<Employees> GetData()
        {
            List<Employees> data = new List<Employees>
            {
                new Employees() { Name = "Andrew Fuller", ID = "1" },
                new Employees() { Name = "Anne Dodsworth", ID = "2" },
                new Employees() { Name = "Janet Leverling", ID = "3" },
                new Employees() { Name = "Laura Callahan", ID = "4" },
                new Employees() { Name = "Margaret Peacock", ID = "5" }
            };
            return data;
        }
    }
}

```


Customize the suggestion item count

While filtering, you can customize the number of list items to be displayed in the suggestion list using the [SuggestionCount](#) property.

CSHTML

```

@model List<string>
@{
    char nameMentionChar = '@';
    List<EmailData> data = new EmailData().EmailList();
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag user"></div>
@Html.EJS().Mention("SuggestionCount").MentionChar(nameMentionChar).Target("#mentionElement").SuggestionCount(8).DataSource((IEnumerable<object>)data).Fields(new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Name" }).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
    }
</style>

```

```

border: 1px solid #D7D7D7;
border-radius: 4px;
padding: 8px;
font-size: 14px;
width: 600px;
}
</style>

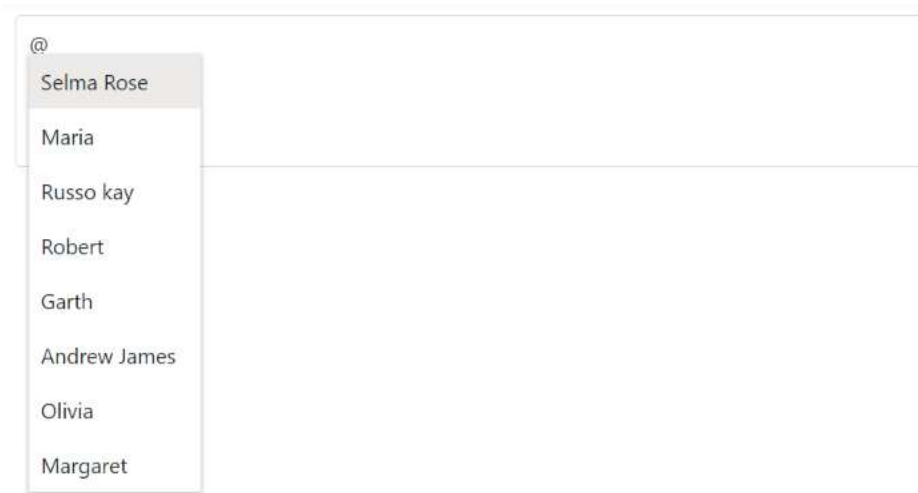
```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class EmailData
    {
        public string Name { get; set; }
        public string EmailId { get; set; }
        public List<EmailData> EmailList()
        {
            List<EmailData> email = new List<EmailData>()
            {
                new EmailData { Name = "Selma Rose", EmailId =
"selma@gmail.com" },
                new EmailData { Name = "Maria", EmailId = "maria@gmail.com"
},
                new EmailData { Name = "Russo kay", EmailId =
"russo@gmail.com" },
                new EmailData { Name = "Robert", EmailId =
"robert@gmail.com" },
                new EmailData { Name = "Garth", EmailId = "garth@gmail.com"
},
                new EmailData { Name = "Andrew James", EmailId =
"andrew@gmail.com" },
                new EmailData { Name = "Olivia", EmailId =
"olivia@gmail.com" },
                new EmailData { Name = "Margaret", EmailId =
"margaret@gmail.com" },
                new EmailData { Name = "Ursula Ann", EmailId =
"ursula@gmail.com" },
                new EmailData { Name = "Laura Grace", EmailId =
"laura@gmail.com" },
                new EmailData { Name = "Albert", EmailId =
"albert@gmail.com" },
                new EmailData { Name = "William", EmailId =
"william@gmail.com" }
            };
            return email;
        }
    }
}

```



See also

- [Templates](#)

Sorting datasource in Mention

Sort order type

You can display the suggestion list items in a specific order. It has possible types as **Ascending**, **Descending**, and **None** in the [SortOrder](#) property.

- **None** - The data source is not sorted.
- **Ascending** - The data source is sorted in ascending order.
- **Descending** - The data source is sorted in descending order.

CSHTML

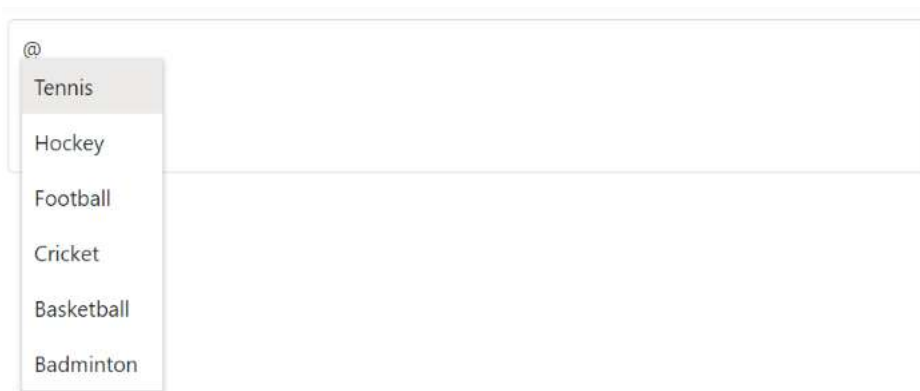
```
@model List<string>
@{
    char nameMentionChar = '@';
    List<SportsData> data = new SportsData().SportsList();
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag sport"></div>
@Html.EJS().Mention("Sorting").MentionChar(nameMentionChar).Target("#mentionElement").Fields(new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Game" , Value = "ID"
}).DataSource((IEnumerable<object>) data).SortOrder(Syncfusion.EJ2.DropDowns.SortOrder.Descending).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
    }
```

```
font-size: 14px;
width: 600px;
}

</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class SportsData
    {
        public string ID { get; set; }
        public string Game { get; set; }
        public List<SportsData> SportsList()
        {
            List<SportsData> sports = new List<SportsData>()
            {
                new SportsData { ID = "game1", Game = "Badminton" },
                new SportsData { ID = "game2", Game = "Football" },
                new SportsData { ID = "game3", Game = "Tennis" },
                new SportsData { ID = "game4", Game = "Hockey" },
                new SportsData { ID = "game5", Game = "Basketball" },
            };
            return sports;
        }
    }
}
```



Templates in Mention

The Mention has been provided with several options to customize each suggestion list item, display item, and data loading indication.

Item template

The content of each list item in the Mention can be customized using the [ItemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data using `ItemTemplate`.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
    var query = "new ej.data.Query().select(['FirstName','Country',
'EmployeeID']).take(6).requiresCount()";
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").PopupWidth("250px").MentionChar(nameMentionC
har).SuggestionCount(6).Target("#mentionElement").ItemTemplate("<span><span
class=\"name\"> ${FirstName}</span><span
class=\"city\"> ${Country}</span></span>").DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor")).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string)query).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    .city {
        right: 15px;
        position: absolute;
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
    }
}
```

```

    {
        return View();
    }
}

```

| | |
|------------------|---------|
| @ | |
| Andrew Fuller | England |
| Anne Dodsworth | USA |
| Janet Leverling | USA |
| Laura Callahan | USA |
| Margaret Peacock | USA |
| Michael Suyama | USA |

Display template

You can customize the mentioned value's display appearance using the [DisplayTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both FirstName and City in the mention element, which is separated by a hyphen.

CSHTML

```

@model List<string>
@{
    char nameMentionChar = '@';
    var query = "new ej.data.Query().select(['FirstName', 'Country',
'EmployeeID']).take(6).requiresCount()";
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").PopupWidth("250px").MentionChar(nameMentionC
har).Target("#mentionElement").ItemTemplate("<span><span class=\"name\">
${FirstName}</span><span
class=\"city\">${Country}</span></span>").DisplayTemplate("<span>${FirstName
} - ${City}</span>").DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor")).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string) query).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    .city {
        right: 15px;
        position: absolute;
    }
</style>
#mentionElement {

```

```

        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>

```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}

```



No records template

You can show the custom design of the popup list content when no data and matches are found on the search with the help of [NoRecordsTemplate](#) property.

CSHTML

```

@model List<string>
@{
    char nameMentionChar = '@';
    string[] data = new string[] { };
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag user"></div>

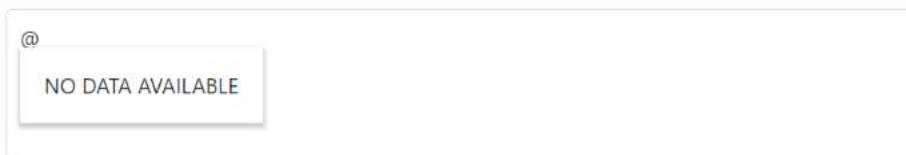
```



```
@Html.EJS().Mention("noRecord-
template").MentionChar(nameMentionChar).NoRecordsTemplate("<span
class='norecord'> NO DATA
AVAILABLE</span>").Target("#mentionElement").DataSource((IEnumerable<object>
)data).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement {
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        {
            public ActionResult Index()
            {
                return View();
            }
        }
    }
}
```



Spinner template

Display the customized waiting spinner, when data fetching takes time to load in the suggestion list by using the [SpinnerTemplate](#) property.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
```

```

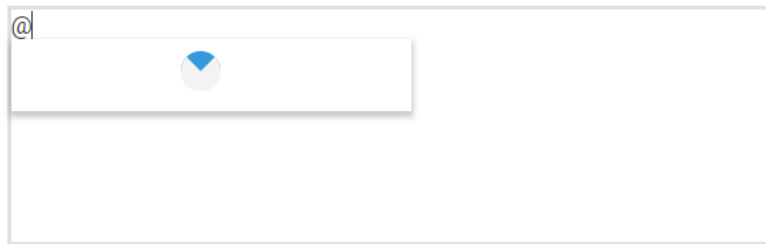
var query = "new
ej.data.Query().select(['FirstName','Country','EmployeeID']).take(6).require
sCount();"
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
user"></div>
@Html.EJS().Mention("employee").MentionChar(nameMentionChar).Target("#mentio
nElement").PopupWidth("200px").SpinnerTemplate("<div
class='spinner_loader'></div>").DataSource(obj =>
obj.Url("https://ej2services.syncfusion.com/production/web-
services/api/Employees").CrossDomain(true).Adaptor("WebApiAdaptor")).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings
{
    Text = "FirstName",
    Value = "EmployeeID"
}).Query((string)query).Render()
<style>
div#mentionElement[placeholder]:empty:before {
    content: attr(placeholder);
}
#mentionElement {
    min-height: 100px;
    border: 1px solid #D7D7D7;
    border-radius: 4px;
    padding: 8px;
    font-size: 14px;
    width: 600px;
}
.spinner_loader {
    border: 10px solid #f3f3f3;
    border-radius: 50%;
    border-top: 10px solid #3498db;
    width: 5px;
    height: 5px;
    position: absolute;
    top: 6px;
    left: 85px;
    -webkit-animation: spin 2s linear infinite; /* Safari */
    animation: spin 2s linear infinite;
}
/* Safari */
@@-webkit-keyframes spin {
    0% {
        -webkit-transform: rotate(0deg);
    }
    100% {
        -webkit-transform: rotate(360deg);
    }
}
@@keyframes spin {
    0% {
        transform: rotate(0deg);
    }
    100% {
        transform: rotate(360deg);
    }
}
}

```

```
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MentionController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```



See also

- [How to achieve filtering](#)

Localization in ASP.NET MVC Mention

ASP.NET MVC Mention control can be localized. Refer to ASP.NET Core [Localization](#) topic to localize Syncfusion ASP.NET MVC controls.

See also

- [Accessibility](#)

Customization in Mention

Show or hide mention character

You can show the mention character as the prefix of the selected item in mention control using [ShowMentionChar](#) property. The default value of `ShowMentionChar` is `false`.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '#';
```

```

        List<EmailData> data = new EmailData().EmailList();
    }
    <div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag
    user"></div>
    @Html.EJS().Mention("Mention-Char-
    Customize").Target("#mentionElement").MentionChar(nameMentionChar).DataSourc
    e((IEnumerable<object>)data).Fields(new
    Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Name"
    }).ShowMentionChar(true).Render()
    <style>
        div#mentionElement[placeholder]:empty:before {
            content: attr(placeholder);
        }
        #mentionElement{
            min-height: 100px;
            border: 1px solid #D7D7D7;
            border-radius: 4px;
            padding: 8px;
            font-size: 14px;
            width: 600px;
        }
    </style>

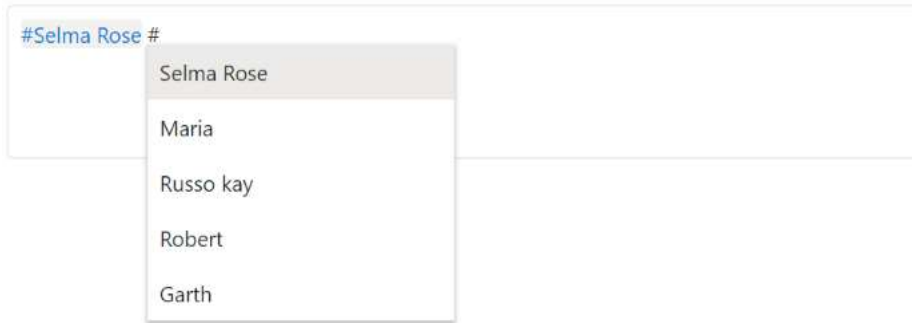
```

DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class EmailData
    {
        public string Name { get; set; }
        public string EmailId { get; set; }
        public List<EmailData> EmailList()
        {
            List<EmailData> email = new List<EmailData>()
            {
                new EmailData { Name = "Selma Rose", EmailId =
                "selma@gmail.com" },
                new EmailData { Name = "Maria", EmailId = "maria@gmail.com"
            },
                new EmailData { Name = "Russo kay", EmailId =
                "russo@gmail.com" },
                new EmailData { Name = "Robert", EmailId =
                "robert@gmail.com" },
                new EmailData { Name = "Garth", EmailId = "garth@gmail.com"
            }
            };
            return email;
        }
    }
}

```



Adding the suffix character after selection

You can add the suffix character while selecting an item in the Mention control using [SuffixText](#) property. You can add space or new line as suffix to the selected item. The default values are empty string.

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
    List<SportsData> data = new SportsData().SportsList();
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag sport"></div>
@Html.EJS().Mention("Suffix").Target("#mentionElement").MentionChar(nameMentionChar).DataSource((IEnumerable<object>)data).Fields(new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Game" , Value = "ID" }).SuffixText("&#160;").Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class SportsData
    {
        public string ID { get; set; }
    }
}
```

```

public string Game { get; set; }
public List<SportsData> SportsList()
{
    List<SportsData> sports = new List<SportsData>()
    {
        new SportsData { ID = "game1", Game = "Badminton" },
        new SportsData { ID = "game2", Game = "Football" },
        new SportsData { ID = "game3", Game = "Tennis" },
        new SportsData { ID = "game4", Game = "Hockey" },
        new SportsData { ID = "game5", Game = "Basketball" },
    };
    return sports;
}
}

```

Configure the popup list

You can customize the suggestion list as width and height using the [PopupHeight](#) and [PopupWidth](#) properties.

By default, the popup list width value is set as `auto`. Depending on the mentioned suggestion data list, the width value is automatically adjusted. The popup list height value is set as `300px`.

CSHTML

```

@model List<string>
@{
    List<Countries> data = new Countries().GetData();
    char nameMentionChar = '@';
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag country"></div>
@Html.EJS().Mention("Popup-Litst-Customize").Target("#mentionElement").MentionChar(nameMentionChar).DataSource((IEnumerable<object>)data).Fields(new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Country" }).PopupWidth("250px").PopupHeight("200px").Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>

```

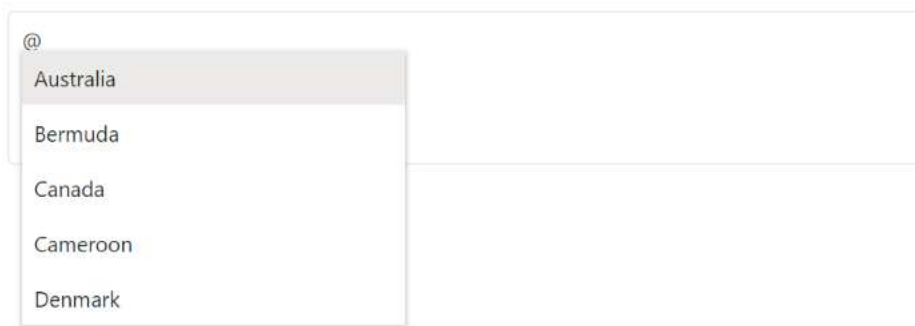
DATA.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Countries
    {
        public string Country { get; set; }
        public string Code { get; set; }
        public List<Countries> GetData()
        {
            List<Countries> data = new List<Countries>()
            {
                new Countries() { Country = "Australia", Code = "AU" },
                new Countries() { Country = "Bermuda", Code = "BM" },
                new Countries() { Country = "Canada", Code = "CA" },
                new Countries() { Country = "Cameroon", Code = "CM" },
                new Countries() { Country = "Denmark", Code = "DK" }
            };
            return data;
        }
    }
}

```



Trigger character

You can customize the trigger character by using the [MentionChar](#) property in the Mention control. The trigger character triggers the suggestion list to display in the target area.

By default, the [MentionChar](#) is @.

Accessibility in Mention

Web accessibility makes web contents and applications more accessible to people with disabilities. The Mention control provides built-in compliance with WAI-ARIA specifications. The WAI-ARIA support is achieved using attributes such as `aria-selected` and `aria-activedescendent`.

The Mention component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Mention component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

| [Right-To-Left Support](#) | |

| [Color Contrast](#) | |

| [Mobile Device Support](#) | |

| [Keyboard Navigation Support](#) | |

| [Accessibility Checker Validation](#) | |

| [Axe-core Accessibility Validation](#) | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

The Mention control uses the **Listbox** role where each list item has an **option** role. The following **ARIA attributes** denote the Mention state.

| **Properties** | **Functionalities** |

| --- | --- |

| aria-selected | Indicates the selected option. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the Mention without interruptions.

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Arrow Down | Selects the first item in the Mention list. Otherwise, selects the item next to the currently selected item. |

| Arrow Up | Selects the item previous to the currently selected one. |

| Esc(Escape) | Closes the popup list when it is in an open state. |

| Enter | Selects the focused item, and when it is in an open state the popup list closes. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, inserts the selected popup list item and closes the popup list. |

CSHTML

```
@model List<string>
@{
    char nameMentionChar = '@';
    List<Employees> data = new Employees().GetData();
}
<div id="mentionElement" placeholder="Type @Html.Raw("mention") and tag employee"></div>
@Html.EJS().Mention("Accessibility").Target("#mentionElement").MentionChar(nameMentionChar).DataSource((IEnumerable<object>)data).Fields(new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Name" }).Render()
<style>
    div#mentionElement[placeholder]:empty:before {
        content: attr(placeholder);
    }
    #mentionElement{
        min-height: 100px;
        border: 1px solid #D7D7D7;
        border-radius: 4px;
        padding: 8px;
        font-size: 14px;
        width: 600px;
    }
</style>
```

DATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Employees
    {
    }
```

```
public string Name { get; set; }
public string ID { get; set; }
public List<Employees> GetData()
{
    List<Employees> data = new List<Employees>
    {
        new Employees() { Name = "Andrew Fuller", ID = "1" },
        new Employees() { Name = "Anne Dodsworth", ID = "2" },
        new Employees() { Name = "Janet Leverling", ID = "3" },
        new Employees() { Name = "Laura Callahan", ID = "4" },
        new Employees() { Name = "Margaret Peacock", ID = "5" }
    };
    return data;
}
}
```

Ensuring accessibility

The Mention component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Mention component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Mention component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Menu

Getting Started with ASP.NET MVC Menu Control

This section briefly explains about how to include [ASP.NET MVC Menu](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in nuget.org. Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Menu control

Now, add the Syncfusion ASP.NET MVC Menu control in `~/Views/Home/Index.cshtml` page.

CSHTML

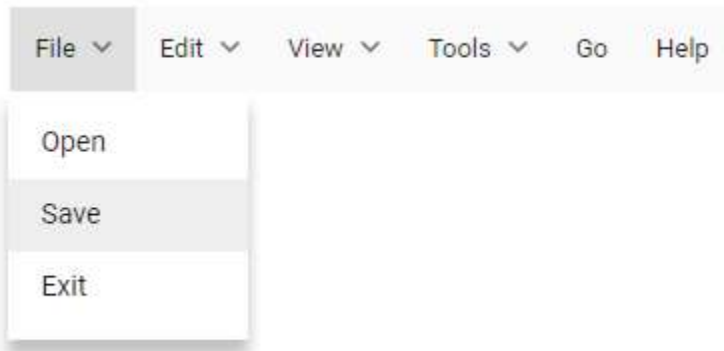
```
@model List<object>
@Html.EJS().Menu("menu").Items(Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "File",
        items = new List<object>()
        {
            new { text = "Open" },
            new { text = "Save" },
            new { text = "Exit" }
        }
    });
    menuItems.Add(new
    {
        text = "Edit",
        items = new List<object>()
        {
            new { text = "Cut" },
            new { text = "Copy" },
            new { text = "Paste" }
        }
    });
    menuItems.Add(new
    {
        text = "View",
        items = new List<object>()
        {
            new { text = "Toolbar" },
            new { text = "Sidebar" },
            new { text = "Fullscreen" }
        }
    });
    menuItems.Add(new
    {
        text = "Tools",
        items = new List<object>()
        {
            new { text = "Spelling & Grammar" },
            new { text = "Customize" },
            new { text = "Options" }
        }
    });
    menuItems.Add(new
    {
        text = "Go"
```

```
});  
menuItems.Add(new  
{  
text = "Help"  
});  
return View(menuItems);  
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Menu control will be rendered in the default web browser.



Note: This example demonstrates the basic rendering of Menu with items support. For more information about data source support, refer to the [Data Source Binding](#) section.

Group menu items with separator

The separators are both horizontal and vertical lines used to separate the menu items. You cannot select the separators, but you can enable separators to group the menu items using the [Separator](#) property.

The **Open** and **Save** sub menu items are grouped using the **separator** property in the following sample.

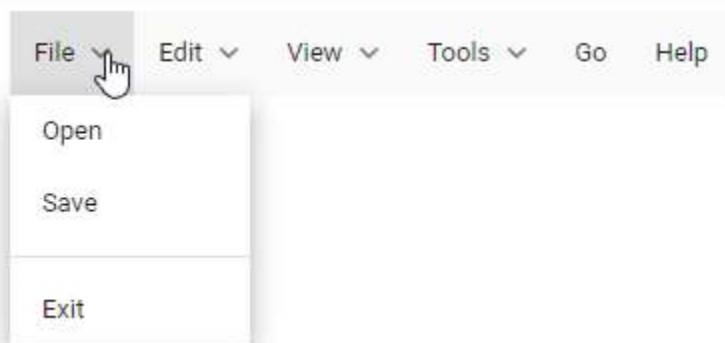
CSHTML

```
@model List<object>  
@Html.EJS().Menu("menu").Items(Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()  
{  
List<object> menuItems = new List<object>();  
menuItems.Add(new  
{  
text = "File",  
items = new List<object>()  
{  
new { text = "Open" },  
new { text = "Save" },  
new { separator = true },  
new { text = "Exit" }  
}  
});  
}
```

```
menuItems.Add(new
{
    text = "Edit",
    items = new List<object>()
    {
        new { text = "Cut" },
        new { text = "Copy" },
        new { text = "Paste" }
    }
});
menuItems.Add(new
{
    text = "View",
    items = new List<object>()
    {
        new { text = "Toolbar" },
        new { text = "Sidebar" },
        new { text = "Fullscreen" }
    }
});
menuItems.Add(new
{
    text = "Tools",
    items = new List<object>()
    {
        new { text = "Spelling & Grammar" },
        new { text = "Customize" },
        new { text = "Options" }
    }
});
menuItems.Add(new
{
    text = "Go"
});
menuItems.Add(new
{
    text = "Help"
});
return View(menuItems);
}
```



Note: The `separator` property should not be given along with the other fields in the `MenuItem`. You can also enable the separator to group **horizontal** menu items.

Note: [View Sample in GitHub.](#)

See also

- [Create menu using data source](#)
- [Customize menu items using template support](#)
- [Integrating with Toolbar component](#)

Icons and Sub menu items

Icons

The menu item contains an icon/image in it to provide a visual representation of an action. To place the icon on a menu item, set the [IconCss](#) property with the required icon CSS. By default, the icon is positioned at the left of the menu item. In the following sample, the icons of File and Edit menu items and Open, Save, Cut, Copy, and Paste sub menu items are added using the `iconCss` property.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()

<style>
/**
 * ej2 Menu styles
 */
@@font-face {
    font-family: 'em-icons';
    src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1tSfgAAAEoAAAVmNtYXdNHzdzAAABoAAAAEJnbHlmAzZ
KdAAAAfGAAAboaGVhZBRYRHEAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAAYAAAAA
gbG9jYQeEBT4AAAHkAAAAEm1heHABFwE+AAABCAAAACBuYw1l1l/aHiQAACOAAAAIxcG9zdIKLcFs
AAAsUAAAAewABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAABAAAAAQAAloT+RV8
PPPUACwQAAAAANii/8AAAAAA2KL/wAAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAIATiABQAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAAABAAAAAQAAAAEAAAAABAAAAAQAAAAAAACAAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAAEAAgADAAQABQAGAAcAAAAAAAAAAbADqAQIBPAKiAxgDdAAAAAU
AAAAA5YD8wADAAcACwAPAFMAADchNSE1ITUhNSE1ISURIREnERUfDTMhMz8NNRE1Lw0jiSMPDeQ
CM/3NAjP9zQIz/c0CdP1QPgEDAqQEBQUGBwYIBwgJCAKGCakIBwgGBwYFBQQEAgMBAQMCBAQFBQY
HBggHCAkI/XoICQgHCAYHBgUFBAQCAWHIPn0/ft59/VECr6f8vvgICAgHBwcGBQUEAwMCAgICAwM
EBQUGBwcHCAGICANCCAGICACHBwYFBQQDAwICAGIDAwQFBQYHBwcHCAkAAAUAAAAA/MD8wACABc
AGQA7AGQAADc5AQc/ATUvDyM5AQkBBHw8BLw43IwcfDz8ENS8LDwP67gHtAgMEBgYICQoLCwwNDQ4
ODwgCgP21Dg8ODg4NDQsLCgkIBgQDAQJLAQIEBgICGoMDA0ODw8PVQE0Dg4PDg0NDQwKCgkIBgQ
DATUMBgIBAQEDBQc/BgcGBgYNCwoKCFxQAU8IDw8ODQ0MDAoKCAgGBgQDAQKB/bYBAwUGBwkKCww
MDg0ODw4OAKoPDg8ODQ4MDAsKCQcGBQNkNQEDBAYICAoLDAwNDg4ODg40FQ4ICQkJBAKCT8EBAI
CagEBAwMEAAAAAAEAAAAAA/QDtQAKAAA3IRMhAxMhNSE1IQwDLLz81JY4A0z+K/4rSgJS/LECDV5
eAAQAAAAAA/QD8wADAAsAGQAjAAABESERARUZnTMVITUjESERMxUzESMRIREjESMRFSERizUjNSE
DHv3EAR5HSP6bSAH0j0dH/TZIRwPor0j8pwFfx/uIBHgI8j4/X1/7iAR5I/O4Bzv6aA1r8pkcDWUh
HAAAABQAAAAAD9AOvAD8ARwBPAI8BMQAAARUPdi8OPQE/Dh8OBQ8DJyU3CQEnAtczHwEFFQ8OLw4
9AT8OHw4FHxAPER8PPw8vDzcBHwI/CS0BLwkPAQEnPw8vDw8OATcCAwQFBgcHCQkKCgsLDAwMDAs
LCgoJCQcHBgUEAwICAwQFBgcHCQkKJCwsLDAwMDAsLCgoJCAgHBgUEAwICWakJCQ8Q/q0mAWb+Nyg
BtwYTCwv9tAIDBAUGBwcJCQoKCwsMDAwMCwsKCgkJBwcGBQQDAgIDBAUGBwcJCQkLCwsMDAwMCws
KCgkJBwcGBQQDAv7VAQIDBQYHCAkLCwsNDg4OFX99CA8ODw0NDAsKCQgHBgUDAgEBAwQGCAkKDA0
ODw8REIRITEHISEBAPDg0LCwkHBwQDAQEBAWMEBQHUHBwQICQoKCxBVAdSiXcguDhAQEAghCAge/nE
Bj44ICAgIEBAQLEHUTEF4fVhELCgoJCAgHBwUFBAQDAwQBAwQBHwLcW0ODxAQEhISEXIREQ8PDg0
MCgkIBgQDAQQLDAsLCgoJCQcHBgUEAwEBAQEDBAUGBwcJCQoKCwsMCwwMCwsKCgkJBwcGBQQDAQE
BAQMEBQYHBwkJCqoLCwwLBAICAQHfFwEe/u8YAQEBAQMCDawLCwoKCQkHBwYFBAMBAQEBAwQFBgc
```

```

HCQkKCgsLDAwLDAsLCgoJCQcHBgUEAwEBAQEDBAUGBwcJCQoKCwsMCxEPEA4ODg0MCwsJCAgGBQV
KSgEEBQYICAKLCwwNDg4OEA8REhIREQ8PDg0LCwkHBgUDAQEDBQYHCQsLDQ4PDxEREhIODQwNDAs
MCgsJCggIBwcIMv7qBAIBAQIEBgkFBgCHJe3uJAghBgUIBgQDAQED/ucyCQcHCAgKCQsKDAsMDQw
NDhISEREpdw4NCwsJBWYFAwEBawUGBwkLCw0ODw8RERIAAAAEAAAAAOWA/QAEAATABkAWQAAARE
hNSE3Mz8HNREhIzclESERMzcFERUFbZMXMx0BHwgZITM/CDURNS8HIycjPQEvCCMhA1j97AF3BwY
GCwoJBgUCAf5LcXEBdv3t2wH+5gECBAcICgWGBgZeAQIFBgkKCWYGBgIUBgYGDaoIBwQCAQECBAc
ICgWGBgZeAQIFBgkKCWYGB/6cAxn9MV4BAgUGCQoLBgYHAjJxLP0xAfTarv3gBwYGCwoJBgUCAV4
GBgYLCgkGBQIBAQIFBgkKCWYGBgLPBwYGCwoJBgUCAV4GBgYLCgkGBQIBAAMAAAAAA4YD9AAHAB4
ARwAAEzMVITUzESEBFTMVITUzNT8HHwYnIxUjESERIZUjLw4rAQ8NuFoB11/9cAF+df6ldQEDBgY
JCQsLDAoKCAcFBKt4mQMonnkDAwUFBQcGCACJCAkKCQoLCgoJCgkICQgHBwYFBgQEazh9ff0SAzg
ebW0eCwoJCAcFAwEBawUHCakKEij8lQNrKQkICAgIBwYGBQUEBAICAgICAgQEBQUGBgCICAgJAAA
AABIA3gABAAAAAAAEAAAAABAAAAAABAAGAAQABAAAAAAACAACACQABAAAAAADAAgAEABAAAA
AAAAEAAGAGAABAAAAAAFAAsAIAABAAAAAAGAAGAKwABAAAAAAAKACwAMwABAAAAAALABIAxWA
DAAEECQAAAAIacQADAAEECQABABAAcWADAAEECQACAA4AgwADAAEECQADABAAkQADAAEECQAEABA
AoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFgAlwADAAEECQALACQBLyBlbS1pY29
uc1JlZ3VsYXJlbS1pY29uc2VtLW1jb25zVmVyc2lvbiAxLjBlbS1pY29uc0ZvbnQgZ2VuZXJhdGV
kIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQBtAC0
AaQBJAG8AbgBzAFIAZQBnAHUAbABhAHIAZQBtAC0AaQBJAG8AbgBzAGUAbQAtAGkAYwBvAG4AcwB
WAGUAcgBzAGkAbwBuACAAMQAuADAAZQBtAC0AaQBJAG8AbgBzAEYAbwBuAHQAIAIBnAGUAbgBlAHIA
AYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuaGMAZgBlAHMAaQBvAG4AIAIBNAGUAdABYAG8AIAIB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGBAGEDAQQBBQEGAQCBCAEJAAXmaWxlLXRleHRfMDE
HZWRpdF8wNQxmaWxlLW9wZW5fMDEHc2F2ZV8wMgZjdXQtd2YHY29weS13ZgxjbGlwYm9hcmQtd2Y
AAAA=) format('trueType');
    font-weight: normal;
    font-style: normal;
}
.em-icons {
    font-family: 'em-icons';
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    line-height: 1;
    text-transform: none;
}
.e-file::before {
    content: '\e700';
}
.e-edit::before {
    content: '\e701';
}
.e-menu-wrapper .e-tool::before {
    content: '\e7cf';
}
.e-menu-wrapper .e-cut::before {
    content: '\e704';
}
.e-menu-wrapper .e-copy::before {
    content: '\e705';
}
.e-menu-wrapper .e-paste::before {
    content: '\e706';
}
.e-menu-wrapper .e-open::before {
    content: '\e702';
}
.e-menu-wrapper .e-save::before {

```



```

        content: '\e703';
    }
</style>

```

ICON.CS

```

// GET: Icons
public ActionResult IconFeatures()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "File",
        iconCss = "em-icons e-file",
        items = new List<object>()
        {
            new { text = "Open", iconCss= "em-icons e-open" },
            new { text = "Save", iconCss= "em-icons e-save" },
            new { separator = true },
            new { text = "Exit" }
        }
    });
    menuItems.Add(new
    {
        text = "Edit",
        iconCss = "em-icons e-edit",
        items = new List<object>()
        {
            new { text = "Cut", iconCss= "em-icons e-cut" },
            new { text = "Copy", iconCss= "em-icons e-copy" },
            new { text = "Paste", iconCss= "em-icons e-paste" }
        }
    });
    menuItems.Add(new
    {
        text = "View",
        items = new List<object>()
        {
            new { text = "Toolbar" },
            new { text = "Sidebar" },
            new { text = "Fullscreen" }
        }
    });
    menuItems.Add(new
    {
        text = "Tools",
        items = new List<object>()
        {
            new { text = "Spelling & Grammar" },
            new { text = "Customize" },
            new { text = "Options" }
        }
    });
    menuItems.Add(new
    {
        text = "Go"
    });
}

```

```

});
menuItems.Add(new
{
    text = "Help"
});
ViewBag.menuItems = menuItems;
return View();
}

```

Navigation

Navigation in Menu is used to navigate to the other web page when a menu item is clicked. It can be achieved by providing a link to the menu item using the url property. In the following sample, the Navigation URL is added to sub menu items using the url property.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeItemRender("beforeItemRender").Render()
<script>
function beforeItemRender(args) {
    if (args.item.url) {
        // To open url in blank page.
        args.element.getElementsByTagName('a')[0].setAttribute('target',
        '_blank');
    }
}
</script>
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>

```

NAVIGATION.CS

```

// Menu Navigation
public ActionResult MenuNavigation()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "Appliances",
        items = new List<object>()
        {
            new { text= "Washing Machine", url=
            "https://www.google.com/search?q=washing+machine"},
            new {text= "Air Conditioners", url=
            "https://www.google.com/search?q=air+conditioners" }
        }
    });
    menuItems.Add(new
    {
        text = "Mobile",
        items = new List<object>()
    }

```

```

        {
            new { text= "Headphones", url=
"https://www.google.com/search?q=headphones" },
            new { text= "Memory Cards", url=
"https://www.google.com/search?q=memory+cards" },
            new { text= "Power Banks", url=
"https://www.google.com/search?q=power+banks" }
        }
    });
    menuItems.Add(new
    {
        text = "Entertainment",
        items = new List<object>()
        {
            new { text= "Televisions", url=
"https://www.google.com/search?q=televisions" },
            new { text= "Home Theatres", url=
"https://www.google.com/search?q=home+theatres" },
            new { text= "Gaming Laptops", url=
"https://www.google.com/search?q=gaming+laptops" }
        }
    });
    menuItems.Add(new
    {
        text = "Fashion",
        url = "https://www.google.com/search?q=fashion"
    });
    menuItems.Add(new
    {
        text = "Offers",
        url = "https://www.google.com/search?q=offers"
    });
    ViewBag.menuItems = menuItems;
    return View();
}

```

Multilevel nesting

The Menu supports multiple level nesting, and it can be achieved by mapping the [Items](#) property inside the parent `menuItem`s.

In the following sample, three-level nesting of menu has been provided.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>

```

NESTING.CS

```

// Menu Multi Level
public ActionResult MultiLevelNesting()

```

```

{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "Fashion",
        items = new List<object>()
        {
            new {
                text = "Men Fashion",
                items = new List<object>()
                {
                    new {
                        text = "Personal Care",
                        items = new List<object>()
                        {
                            new { text = "Trimmers" },
                            new { text = "Shavers" }
                        }
                    },
                    new {
                        text = "Clothing",
                        items = new List<object>()
                        {
                            new { text = "Shirts" },
                            new { text = "Jackets" },
                            new { text = "Track Suits" }
                        }
                    },
                    new { text = "Footwear" }
                }
            },
            new {
                text = "Women Fashion",
                items = new List<object>()
                {
                    new {
                        text = "Clothing",
                        items = new List<object>()
                        {
                            new { text = "Kurtas" },
                            new { text = "Salwars" },
                            new { text = "Sarees" }
                        }
                    },
                    new {
                        text = "Jewellery",
                        items = new List<object>()
                        {
                            new { text = "Nosepins" },
                            new { text = "Anklets" }
                        }
                    }
                }
            }
        }
    });
    menuItems.Add(new

```

```

{
    text = "Home & Living",
    items = new List<object>()
    {
        new {
            text = "Washing Machine",
            items = new List<object>()
            {
                new { text = "Fully Automatic" },
                new { text = "Semi Automatic" }
            }
        },
        new {
            text = "Air Conditioners",
            items = new List<object>()
            {
                new { text = "Inverter ACs" },
                new { text = "Split ACs" }
            }
        }
    }
});
menuItems.Add(new
{
    text = "Accessories",
});
menuItems.Add(new
{
    text = "Sports",
});
menuItems.Add(new
{
    text = "Gaming",
});
ViewBag.menuItems = menuItems;
return View();
}

```

Note: You can achieve multi level nesting with data source by mapping **name** of the child items to the **children** sub-property of **fields** property. For more information, refer to the [data source binding](#) section.

The below table represents the MenuItem properties and its description.

| Property Name | Type | Description |
|---------------|-----------------|--|
| iconCss | string | Defines class/multiple classes separated by a space for the menu Item that is used to include an icon. Menu Item can include font icon and sprite image. |
| id | string | Specifies the id for menu item. |
| separator | boolean | Specifies separator between the menu items. Separator are either horizontal or vertical lines used to group menu items. |
| items | MenuItemModel[] | Specifies the sub menu items that is the array of MenuItem model/ |
| text | string | Specifies text for menu item. |

|url|string|Specifies url for menu item that creates the anchor link to navigate to the url provided.

See Also

- [Customize menu items](#)
- [Group menu items with separator](#)

Data source binding and Custom menu items

Data binding

The Menu supports data source bindings such as array of JavaScript objects that can be structured as either **hierarchical** or **self-referential** data.

Hierarchical data

The Menu can be populated with hierarchical data source by assigning it to the **items** property, and the fields with corresponding keys can be mapped to the [fields](#) property.

JSON data

The Menu can generate its menu items through an array of complex data source by mapping fields from the [fields](#) property.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Fields(ViewBag.menuFields)
.Render()
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>
```

MODEL.CS

```
public class Model
{
    public static List<object> GetLocalData()
    {
        List<object> data = new List<object>()
        {
            new
            {
                continent = "Asia",
                countries = new List<object>()
                {
                    new
                    {
                        country = "China",
                        languages = new List<object>()
                        {
                            new { language= "Chinese"},
                            new { language= "Cantonese"}
                        }
                    },
                    new {
```

```

        country = "India",
        languages =new List<object>() {
            new { language= "English"},
            new { language= "Hindi"},
            new { language= "Tamil"}
        },
        new {
            country = "Japan",
            languages =new List<object>() {
                new { language= "Japanese"}
            }
        }
    },
    new
    {
        continent = "Africa",
        countries = new List<object>()
        {
            new {
                country = "Nigeria",
                languages =new List<object>() {
                    new { language= "English"},
                    new { language= "Hausa"}
                }
            },
            new
            {
                country = "Egypt",
                languages =new List<object>() {
                    new { language= "Arabic"}
                }
            },
            new
            {
                country = "South Africa",
                languages =new List<object>() {
                    new { language= "Tswana"},
                    new { language= "Swati"}
                }
            }
        }
    },
    new
    {
        continent = "North America",
        countries = new List<object>()
        {
            new
            {
                country = "Canada",
                languages =new List<object>() {
                    new { language= "French"}
                }
            },
            new

```

```

        {
            country = "Mexico",
            languages =new List<object>() {
                new { language= "Spanish"}
            }
        },
        new
        {
            country = "USA",
            languages =new List<object>() {
                new { language= "English"}
            }
        }
    },
    new
    {
        continent = "South America",
        countries = new List<object>()
        {
            new
            {
                country = "Brazil",
                languages =new List<object>() {
                    new { language= "Portuguese"}
                }
            },
            new
            {
                country = "Colombia",
                languages =new List<object>() {
                    new { language= "Spanish"}
                }
            },
            new
            {
                country = "Argentina",
                languages =new List<object>() {
                    new { language= "Spanish"},
                }
            }
        }
    },
    new
    {
        continent = "Oceania",
        countries = new List<object>()
        {
            new { country = "Australia"},
            new { country = "New Zealand"},
            new { country = "Samoa"}
        }
    }
};
return data;
}
}

```


Self-referential data

Menu can be populated from self-referential data structure that contains array of JSON objects with `parentId` mapping.

You can directly assign self-referential data to the `items` property, and map all the field members with corresponding keys from self-referential data to `fields` property.

To render the root level nodes, specify the `parentId` as null or no need to specify the `parentId` in data source.

In the following example, `id`, `pId`, and `text` columns from self-referential data have been mapped to `itemId`, `parentId`, and `text` fields, respectively.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Fields(ViewBag.menuFields)
.Render()
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>
```

SELFREFERENTIAL.CS

```
// Self Referential Data
public ActionResult SelfReferential()
{
    List<object> menuItems = new List<object>()
    {
        new { id = "parent1", text = "Events" },
        new { id = "parent2", text = "Movies" },
        new { id = "parent3", text = "Directory" },
        new { id = "parent4", text = "Queries", pId = "null" },
        new { id = "parent5", text = "Services", pId = "null" },
        new { id = "parent6", text = "Conferences", parentId =
"parent1" },
        new { id = "parent7", text = "Music", parentId = "parent1"
},
        new { id = "parent8", text = "Workshops", parentId =
"parent1" },
        new { id = "parent9", text = "Now Showing", parentId =
"parent2" },
        new { id = "parent10", text = "Coming Soon", parentId =
"parent2" },
        new { id = "parent10", text = "Media Gallery", parentId =
"parent3" },
        new { id = "parent11", text = "Newsletters", parentId =
"parent3" },
        new { id = "parent12", text = "Our Policy", parentId =
"parent4" },
        new { id = "parent13", text = "Site Map", parentId =
"parent4" },
        new { id = "parent14", text = "Pop", parentId = "parent7" },
    }
```

```

        new { id = "parent15", text = "Folk", parentId = "parent7"
    },
        new { id = "parent16", text = "Classical", parentId =
"parent7" }
    };
    // For MenuFieldSettings type, include Syncfusion.EJ2.Navigations in the
using directive section.
    MenuFieldSettings menuFields = new MenuFieldSettings()
    {
        ItemId = "id",
        Text = "text",
        ParentId = "parentId"
    };
    ViewBag.menuItems = menuItems;
    ViewBag.menuFields = menuFields;
    return View();
}

```

Custom menu items

The Menu can be customized using Essential JS2 [Template engine](#) to render the elements.

To customize menu items in your application, set your customized template string to the [template](#) property.

In the following example, the menu has been rendered with customized menu items.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.data).Fields(ViewBag.menuFields).Template("#menuTemplate").Render()
<script id="menuTemplate" type="text/x-template">
    ${if(category)}
    ${category}
    ${else if (value && url)}
    <div class="e-avatar e-avatar-small image" style="background-image:
url(images/${url}.png);">${value}</div>
    ${else if (support)}
    <ul>
        ${for(val of support)}
        <li>
            ${val.value}
            ${if(val.count)}
            <span class='e-badge e-badge-success'>${val.count}</span>
            ${/if}
        </li>
        ${/for}
    </ul>
    ${else}
    <div tabindex="0" class="e-card">
        <div class="e-card-header">
            <div class="e-card-header-caption">
                <div class="e-card-header-title">About Us</div>
            </div>
        </div>
        <div class="e-card-content">
            ${about.value}

```

```

        </div>
        <div class="e-card-actions">
            <button class="e-btn e-outline">
                Read More
            </button>
        </div>
    </div>
    ${/if}
</script>
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
    .menu-control {
        margin-top: 45px;
        text-align: center;
    }
    /**
    * Common customization
    */
    .e-bigger .e-menu-wrapper.e-control ul.e-ul .e-menu-item,
    .e-menu-wrapper.e-control ul.e-ul .e-menu-item {
        display: flex;
        height: auto;
        padding: 0;
    }
    /**
    * Avatar customization
    */
    .e-bigger .e-menu-wrapper ul .e-menu-item .e-avatar {
        background-size: auto;
        text-indent: 40px;
    }
    .e-menu-wrapper ul .e-menu-item .e-avatar {
        background-color: inherit;
        background-position: 0;
        background-size: 25px;
        color: inherit;
        font-size: inherit;
        height: inherit;
        //justify-content: left;
        margin: 0 10px;
        width: 100%;
        text-indent: 35px;
    }
    /**
    * Badge customization
    */
    .e-menu-wrapper ul .e-menu-item ul li {
        padding: 0 10px;
    }
    .e-bigger .e-menu-wrapper ul .e-menu-item ul li .e-badge {
        margin: 18px 0px 0px 10px;
    }
    .e-menu-wrapper ul .e-menu-item ul li .e-badge {
        float: right;
    }

```

```

        margin: 13px 0px 0px 10px;
    }
    .e-menu-wrapper ul .e-menu-item ul li:hover {
        background-color: #eee;
    }
    .fabric .e-menu-wrapper ul .e-menu-item ul li:hover {
        background-color: #eaeaea;
    }
    .bootstrap .e-menu-wrapper ul .e-menu-item ul li:hover {
        background-color: #e6e6e6;
    }
    .highcontrast .e-menu-wrapper ul .e-menu-item ul li:hover {
        background-color: #685708;
    }
    /**
    * Card customization
    */
    .e-bigger .e-menu-wrapper ul.e-ul .e-menu-item .e-card {
        width: 320px;
    }
    .e-menu-wrapper ul.e-ul .e-menu-item .e-card {
        width: 290px;
        font-size: inherit;
        cursor: default;
        background-color: inherit;
        border-color: transparent;
    }
    .e-menu-wrapper ul.e-ul .e-menu-item .e-card .e-card-content {
        white-space: normal;
        color: inherit;
        padding-top: 0;
        text-align: justify;
        font-size: inherit;
    }
    .e-bigger .e-menu-wrapper ul.e-ul .e-menu-item .e-card .cb-icons {
        width: 40px;
        font-size: 40px;
        height: 40px;
    }
    .e-menu-wrapper ul.e-ul .e-menu-item .e-card .cb-icons {
        width: 30px;
        font-size: 30px;
        height: 30px;
    }
    .e-menu-wrapper ul.e-ul .e-menu-item .e-card .e-card-btn {
        background-color: inherit;
    }
}
</style>

```

TEMPLATE.CS

```

// Custom Menu Items
public ActionResult CustomMenuItems()
{
    List<object> data = new List<object>() {
        new {

```

```

        category = "Products",
        options = new List<object>() {
            new { value= "JavaScript", url= "javascript" },
            new { value= "Angular", url= "angular" },
            new { value= "ASP.NET Core", url= "core" },
            new { value= "ASP.NET MVC", url= "mvc" }
        },
        new {
            category = "Services",
            options = new List<object>() {
                new { value= "Application Development", count= "1200+" },
                new { value= "Maintenance & Support", count= "3700+" },
                new { value= "Quality Assurance" },
                new { value= "Cloud Integration", count= "900+" }
            }
        },
        new {
            category = "About Us",
            options = new List<object>() {
                new {
                    id = "about",
                    about = new List<object>() { new { value = "We are on a mission to provide world-class best software solutions for web, mobile and desktop platforms. Around 900+ applications are desgined and delivered to our customers to make digital & strengthen their businesses." } }[0]
                }
            }
        },
        new { category = "Careers" },
        new { category = "Sign In" }
    };
    MenuFieldSettings menuFields = new MenuFieldSettings()
    {
        Text = new string[] { "category", "value" },
        Children = new string[] { "options" }
    };
    ViewBag.data = data;
    ViewBag.menuFields = menuFields;
    return View();
}

```

See Also

- [Render menu with items](#)

Use case scenarios

Scrollable menu

The menu component supports horizontal and vertical scrolling to render large menus and submenus in an adaptive way. This can be achieved by enabling the [enableScrolling](#) property and by restricting the corresponding menu/submenu size.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).EnableScrolling(true).CssClass('e-scrollable-menu').BeforeOpen('onBeforeOpen').Render()
<script>
    function onBeforeOpen(args) {
        // Restricting sub menu wrapper height
        if (args.parentItem.text === 'Appliances') {
            ej.base.closest(args.element, '.e-menu-wrapper').style.height =
'230px';
        }
    }
</script>
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
    /* Restricting the parent menu wrapper size */
    .e-menu-wrapper.e-scrollable-menu:not(.e-menu-popup) {
        width: 492px;
    }
</style>
```

SCROLLABLE.CS

```
public ActionResult Scrollable()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "Appliances",
        items = new List<object>()
        {
            new {
                text= "Kitchen",
                items = new List<object>()
                {
                    new { text= "Electric Cookers" },
                    new { text= "Coffee Makers" },
                    new { text= "Blenders" },
                    new { text= "Microwave Ovens" }
                }
            },
            new {
                text= "Television",
                items = new List<object>()
                {
                    new { text= "Our Exclusive TVs" },
                    new { text= "Smart TVs" },
                    new { text= "Big Screen TVs" }
                }
            },
            new { text= "Washing Machine" },
            new {
                text= "Refrigerators",
                items = new List<object>()
                {
```

```

        new { text= "Inverter ACs" },
        new { text= "Split ACs" },
        new { text= "Window ACs" }
    },
    new { text= "Air Conditioners" },
    new { text= "Water Purifiers" },
    new { text= "Air Purifiers" },
    new { text= "Chimneys" },
    new { text= "Inverters" },
    new { text= "Healthy Living" },
    new { text= "Vacuum Cleaners" },
    new { text= "Room Heaters" },
    new { text= "New Launches" }
}
});
menuItems.Add(new
{
    text = "Accessories",
    items = new List<object>()
    {
        new {
            text= "Mobile",
            items = new List<object>()
            {
                new { text= "Headphones" },
                new { text= "Memory Cards" },
                new { text= "Power Banks" },
                new { text= "Mobile Cases" },
                new { text= "Screen Protectors" }
            }
        },
        new { text= "Laptops" },
        new {
            text= "Desktop PC",
            items = new List<object>()
            {
                new { text= "Pendrives" },
                new { text= "External Hard Disks" },
                new { text= "Monitors" },
                new { text= "Keyboards" }
            }
        },
        new {
            text= "Camera",
            items = new List<object>()
            {
                new { text= "Lens" },
                new { text= "Tripods" }
            }
        }
    }
});
menuItems.Add(new
{
    text = "Fashion",
    items = new List<object>()

```

```

        {
            new { text = "Men" },
            new { text = "Women" }
        }
    });
    menuItems.Add(new
    {
        text = "Home & Living",
        items = new List<object>()
        {
            new { text= "Furniture" },
            new { text= "Decor" },
            new { text= "Smart Home Automation" },
            new { text= "Dining & Serving" }
        }
    });
    menuItems.Add(new
    {
        text = "Entertainment",
        items = new List<object>()
        {
            new { text= "Televisions" },
            new { text= "Home Theatres" },
            new { text= "Gaming Laptops" }
        }
    });
    menuItems.Add(new
    {
        text = "Contact Us"
    });
    menuItems.Add(new
    {
        text = "Help"
    });
    ViewBag.menuItems = menuItems;
    return View();
}

```

Menu in toolbar

The following example demonstrates how to integrate Menu with Toolbar component.

CSHTML

```

<div class="control-section">
    <div class="toolbar-menu-control">

@Html.EJS().Toolbar("shoppingtoolbar").Created("create").Items(ViewBag.items)
.Render()
    </div>
</div>
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
    @@font-face {

```



```

font-family: 'e-menu';
src:
url (data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjvJSpgAAAEoAAAAVmNtYXBsm2feAAABpAAAAGxnbHlmmEc
yrQAAAIQAAAWIaGVhZBj0bwcAAADQAAAAANmhoZWEHmQNyAAAArAAAACRobXR4I0AAAAAAAYAAAA
kbG9jYQaGB+4AAAIQAAAAAFG1heHABGACaAAABCAAAACBuYW1lc0c0BGAAB6wAAAI1cG9zdJbKd4k
AAAnUAAAAfQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAACQABAAAAQAahka7o18
PPPUACwPoAAAAANe2FRwAAAAA17YVHAAAAAAD6gPqAAAACAACAAAAAAAAAAAAEAAAAJAI4ABQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPrAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5anohgNS/2oAWgPqAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAAAAAAAgAAAAAAAUAAAAQAAABQABABYAAAAADgA
IAAIABuWp5bPluefo6CLohv//AADlqeWy5bnn6Ogi6IX//wAAAAAAAAAAAAAAAAABAA4ADgAQABA
AEAAQAAABQAGAAcACAABAAIAAwAEAAAAACsASoBRAGwAhICUAKEAsQABAAAAAAD6gNZAD8AfWC
DAI0AAAEzHw0dAQ8OLw8/DiMzHw0dAQ8OLw8/DgMhAyEBIRU3EyUVBQMjAwgJCAgIBwCHBgUFBAQ
DAgEBAgMEBAUFBgCHBgICakJCAgIBwCHBgUFBAQDAgEBAQECAwQEBQUGBwCHCAgI5AgJCAgHBwC
GBQUEBAIDAQEDAgQEBQUGBwCHCAgJCAkICAgIBwYGBQUFAwMCAQEBAQIDAUFbQYGBwGICaijAny
Q/qj+EgEKAssBcf5Yy9UBTWICAgQEBQYGBGCHCAgJCAkICQcIBwYGBGQFAwMCAQEBAQIDAUEBgY
GBwGHCQgJCAkICACHBgYGBQQEAgICAgICBAQFBgYGBwCICakICQgJBwGHBgYGBAUDAwIBAQEBAgM
DBQQGBgYHCAcJCAkICQgIBwCGBgYFBAQCAgIBu/67AZUBaf5LAj0CABUAAAAFAAAAAAPqA+oAAgA
WABgAPABkAAA3QEnMx8PFQc3MQE7AR8OAQcvDwEzHwOPBi8PPwP/nAgODg4NDAwLCwoICACFBAM
C6k4CdAgHEA4PDQ0MDAoJCAcGBAIB/kWFAQMEBgCJCgsLDQ0NDg4ODgLaBg0GBgYGBjwFBAMBAQE
CAgYJNAEDBAYHCQoKDAwNDQ4ODg40GQkKZJsBAwQFBwCJCQoMCw0NDg8OCE7pAnUDBQYHCQkLDaw
NDg0ODg7+SIGODg4NDg0MDAsKCAgGBAMBARUCAgMDBQU9CQkJCQgICAcNDjQNDg4ODQ0MDAsJCQc
GBAMBNA4DAgAAAAABAAAAAPqA60ACgAAEYEVIRUhAxMhAyEVAcwBzPzEN5MDHrj84gOtXFz9/QG
n/boAAAAABQAAAAADjgPqAAMABwALAA8AUwAAEYEVITUhFSE1IRUhJxEhESUhHw8RDw8hLw8RPw7
qAij92AIo/dgCKP3YOWKi/XICegGICAgHBwYGBQUEAwMCAQEBAQIDAUFbQYGBwCICAgI/YYICAg
IBwCGBgUFbAMDAGABAQECAwMEBQUGBGCCHCAgIAQs+9j72Prj9XgKi9gEBAgMDBAUFBgYHBwGICaj
8zgGICAgHBwYGBQUEAwMCAQEBAQIDAUFbQYGBwCICAgIAzIICAgIBwCGBgUFbAMDAGABQAAAAA
DqQOpAAQACgAUAB4AOWAACQEXATUBFAcmNDIDBgcuATQ2MhYUAWYHLgE0NjIWFBC2NS4BIgYUfhc
yNxchJiMOARQWmjY3Ncc3ATM1Ayb++FkBMv5fFRUq3xglJjExSzEZGCUmMTFLMUoOAmKUY2JLJyF
mZiEnS2JilWICDmcBM4MDgP74WQE2K/50FQICKv6lGQICMkoyMkoB9xkCAjJKMjJKIyEnSmNj1GM
CDmdnDgJj1GNjSichZ/7NKwAAAAAMAAAAA4oD5gAHABAAJwAAARUHTMTRIRe1HgEGiiY0NjInBgc
jIgYVERQWMyEyNjURNcYrAS4BIgEZAbZd/ZABWAwBgIYZGSZhIg+8JjU1JgJ2JjU1JrWPRFGDLn5
9/TICz1IMJxkZJxLAGSkzJv0pJzMzJwLXJjMpMwADAAAAAOpA+cAAwAUAB4AAAERIREnBhURFBY
XIT4BNRE0JiMhIicGFREzESE1IQYDTP4MQxs2JgH3JzU1J/4JJtgZXQIT/egmAs/9jwJxRBkm/Yc
mMwICMyYCEsYynxon/Y8Cv4CAAIAAAAAA+cD5wALACMAAAEOAQcuASc+ATceAQUeARcyNj8BARY
yNjQnATc+ATUuAScOALYA7SHiLMEBLOIh7T9KwXnrkeBNAMBAQ4kHA7+/wMpLgTora7nAk6HtAM
DtIeIswQEs4it6AQUKQP+/w4bJQ4BAQM0gUeu5wUF5wAAAAASAN4AAQAAAAAAAAABAAAAQAAAAA
AAQAHAAEAQAAAAAAAAgAHAAGAAQAAAAAAAAAwAHAA8AAQAAAAAABAHAHABYAAQAAAAAABQALAB0AAQA
AAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBakAAAACAG0AAwABBakAAQAOAG8
AAwABBakAAgAOAH0AAwABBakAAwAOAISAAwABBakABAAOAJKAwABBakABQAWAKCAwABBakABgA
OAL0AAwABBakACgBYAMsAAwABBakACwAkASMgZS1pY29uc1JlZ3VsYXJlLW1jb25zZS1pY29uc1Z
lcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R
lZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGUALQB
pAGMabwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAZQAtAGkAYwBvAG4
AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEADABLAGQAIAB1AHMAaQBwAGcAIABTAKAbgBjAGYAdQB
zAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGk
AbwBuAC4AYwBvAG0AAAAAAGAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJAQIBAwEEAAQU
BBgEHAQgBCQEKAALzAg9wcGluZy1jYXJ0B2VkaXQtMDUMZmlsZS1vcGVuLTAxDGZpbGtGV4dC0
wMQNDdXQFUGFZdGUGFyY2gAAAAA==) format('trueType');

font-weight: normal;
font-style: normal;
}
.em-icons {
font-family: 'e-menu';
font-style: normal;
font-variant: normal;

```

```

        font-weight: normal;
        text-transform: none;
        line-height: 2;
    }
    .toolbar-menu-control .e-toolbar-items .e-toolbar-item .e-tbar-btn .e-
    btn-icon.e-shopping-cart {
        font-size: 20px;
        margin-right: 1px;
    }
    .toolbar-menu-control .e-shopping-cart::before {
        content: '\e7e8';
    }
    .toolbar-menu-control .e-menu-wrapper .e-menu {
        border: none;
    }
    .toolbar-menu-control {
        margin: 45px auto 0;
        max-width: 950px;
    }
    .toolbar-menu-control .e-toolbar .e-toolbar-left .e-toolbar-item.e-
    template {
        padding: 0;
    }
    .toolbar-menu-control .e-toolbar {
        overflow: visible !important;
    }
    .toolbar-menu-control .e-menu-wrapper {
        margin-right: 160px;
    }
    .toolbar-menu-control .e-hscroll .e-hscroll-content {
        position: static;
    }
</style>
<script>
    var menuItems = [
        {
            header: 'Events',
            subItems: [
                { text: 'Conferences' },
                { text: 'Music' },
                { text: 'Workshops' }
            ]
        },
        {
            header: 'Movies',
            subItems: [
                { text: 'Now Showing' },
                { text: 'Coming Soon' }
            ]
        },
        {
            header: 'Directory',
            subItems: [
                { text: 'Media Gallery' },
                { text: 'Newsletters' }
            ]
        }
    ],

```

```

        {
            header: 'Queries',
            subItems: [
                { text: 'Our Policy' },
                { text: 'Site Map' },
                { text: '24x7 Support' }
            ]
        },
        { header: 'Services' }
    ];
    //Menu fields definition
    var menuFields = {
        text: ['header', 'text', 'value'],
        children: ['subItems', 'options']
    };
    //Menu model definition
    var menuOptions = {
        items: menuItems,
        fields: menuFields
    };
    function create() {
        new ej.navigations.Menu({ items: menuOptions.items, fields:
menuOptions.fields, animationSettings: { effect: 'none' } }, '#menu');
        this.refreshOverflow();
    }
</script>

```

TOOLBAR.CS

```

// ToolBar
public ActionResult ToolBar()
{
    List<ToolBarItem> items = new List<ToolBarItem>()
    {
        new ToolBarItem { Template = "<ul id='menu'></ul>" },
        new ToolBarItem { PrefixIcon = "em-icons e-shopping-cart",
Align=Syncfusion.EJ2.Navigations.ItemAlign.Right }
    };
    ViewBag.items = items;
    return View();
}

```

Hamburger menu

The following example demonstrates the use case of menu with Accordion component integrated in SideBar.

CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Navigations;
<div id='container'>
    <!-- sidebar element declaration -->
    <div class="header">
        <span id="hamburger" class="e-icons menu default"></span>
        <div class="content">Header content</div>
    </div>
</div>

```

```

</div>
@Html.EJS().Sidebar("default-
sidebar").Type(Syncfusion.EJ2.Navigations.SidebarType.Over).Width("220px").C
ontentTemplate(
    @<div>
        <div class="title-header">
            <div style="display:inline-block">Menu</div>
            <span id="close" class="e-icons"></span>
        </div>
        <div class="content-area">
            @Html.EJS().Accordion("accordionMenu").Items(new
List<AccordionAccordionItem> {
                new AccordionAccordionItem { Header= "Appliances",
Content= "#Appliances_Items" },
                new AccordionAccordionItem { Header= "Accessories",
Content= "#Accessories_Items" },
                new AccordionAccordionItem { Header= "Fashion",
Content= "#Fashion_Items" },
                new AccordionAccordionItem { Header= "Home &
Living", Content= "#Home_Living_Items" },
                new AccordionAccordionItem { Header=
"Entertainment", Content= "#Entertainment_Items" }
            }).Clicked("clicked").Render()
            <div style="display:none">
                <!--Accordion sub items-->
                @Html.EJS().Accordion("Appliances_Items").Items(new
List<AccordionAccordionItem> {
                    new AccordionAccordionItem { Header= "Kitchen",
Content="#Appliances_Kitchen_Items" },
                    new AccordionAccordionItem { Header= "Washing
Machine", Content="#Appliances_Washing_Items" },
                    new AccordionAccordionItem { Header= "Air
Conditioners", Content="#Appliances_Conditioners_Items" }
                }).Clicked("clicked").Render()
            </div>
            <div style="display:none">

@Html.EJS().Accordion("Appliances_Kitchen_Items").Items(new
List<AccordionAccordionItem> {
                new AccordionAccordionItem { Header= "Electric
Cookers" },
                new AccordionAccordionItem { Header= "Coffee Makers"
            },
                new AccordionAccordionItem { Header= "Blenders" }
            }).Clicked("clicked").Render()
            </div>
            <div style="display:none">

@Html.EJS().Accordion("Appliances_Washing_Items").Items(new
List<AccordionAccordionItem> {
                new AccordionAccordionItem { Header= "Fully
Automatic" },
                new AccordionAccordionItem { Header= "Semi
Automatic" }
            }).Clicked("clicked").Render()
            </div>
            <div style="display:none">

```

```

@Html.EJS().Accordion("Appliances_Conditioners_Items").Items(new
List<AccordionAccordionItem> {
    new AccordionAccordionItem { Header= "Inverter ACs"
},
    new AccordionAccordionItem { Header= "Split ACs" },
    new AccordionAccordionItem { Header= "Window ACs" }
}).Clicked("clicked").Render()
</div>
<div style="display:none">
    @Html.EJS().Accordion("Accessories_Items").Items(new
List<AccordionAccordionItem> {
    new AccordionAccordionItem { Header= "Mobile",
Content="#Accessories_Mobile_Items" },
    new AccordionAccordionItem { Header= "Computer",
Content="#Accessories_Computer_Items" }
}).Clicked("clicked").Render()
</div>
<div style="display:none">

@Html.EJS().Accordion("Accessories_Mobile_Items").Items(new
List<AccordionAccordionItem> {
    new AccordionAccordionItem { Header= "Headphones" },
    new AccordionAccordionItem { Header= "Memory Cards"
},
    new AccordionAccordionItem { Header= "Power Banks" }
}).Clicked("clicked").Render()
</div>
<div style="display:none">

@Html.EJS().Accordion("Accessories_Computer_Items").Items(new
List<AccordionAccordionItem> {
    new AccordionAccordionItem { Header= "Pendrives" },
    new AccordionAccordionItem { Header= "External Hard
Disks" },
    new AccordionAccordionItem { Header= "Monitors" }
}).Clicked("clicked").Render()
</div>
<div style="display:none">
    @Html.EJS().Accordion("Fashion_Items").Items(new
List<AccordionAccordionItem> {
    new AccordionAccordionItem { Header= "Men" },
    new AccordionAccordionItem { Header= "Women" }
}).Clicked("clicked").Render()
</div>
<div style="display:none">
    @Html.EJS().Accordion("Home_Living_Items").Items(new
List<AccordionAccordionItem> {
    new AccordionAccordionItem { Header= "Furniture" },
    new AccordionAccordionItem { Header= "Decor" }
}).Clicked("clicked").Render()
</div>
<div style="display:none">

@Html.EJS().Accordion("Entertainment_Items").Items(new
List<AccordionAccordionItem> {

```

```

        new AccordionAccordionItem { Header= "Televisions"
    },
        new AccordionAccordionItem { Header= "Home Theatres"
    },
        new AccordionAccordionItem { Header= "Gaming
Laptops" }
    }).Clicked("clicked").Render()
</div>
</div>
</div>).Render()
<!-- main content declaration -->
<div>
    <div class="main-content">Main content</div>
</div>
</div>
<style>
    body {
        margin: 0;
    }
    .header {
        width: 100%;
        background-color: #7b8cfb;
    }
    #default-sidebar,
    .header .content {
        background-color: #7b8cfb;
        color: white;
        border: none;
    }
    .header .content {
        font-size: 20px;
        line-height: 50px;
        text-align: center;
    }
    .main-content {
        text-align: center;
        font-size: 20px;
        padding: 100px 15px;
    }
    #default-sidebar .close-btn:hover {
        color: rgba(0, 0, 0, .87);
        background-color: #fafafa;
    }
    #hamburger.menu {
        font-size: 25px;
        cursor: pointer;
        float: left;
        line-height: 50px;
        position: relative;
        z-index: 1000;
        left: 25px;
        color: white;
    }
    #hamburger.menu:before {
        content: '\e99a';
    }
    #close:before {

```

```

        content: "\e945";
    }
    .title-header {
        text-align: center;
        font-size: 18px;
        padding: 15px 15px 35px;
    }
    .e-sidebar .title-header #close {
        cursor: pointer;
        line-height: 25px;
        float: right;
    }
    .e-accordion .e-acrdn-item .e-acrdn-panel .e-acrdn-content {
        padding: 0px;
    }
    .e-accordion .e-acrdn-item .e-acrdn-panel .e-acrdn-panel .e-acrdn-
content {
        padding: 0px;
    }
</style>
<script>
    // Close the Sidebar
    document.getElementById('close').onclick = function () {
        var defaultSidebar =
ej.base.getComponent(document.getElementById("default-sidebar"), "sidebar");
        defaultSidebar.hide();
    };
    document.getElementById('hamburger').onclick = function () {
        var defaultSidebar =
ej.base.getComponent(document.getElementById("default-sidebar"), "sidebar");
        defaultSidebar.show();
    };
    function clicked(e) {
        if (!(e.originalEvent.target.closest('.e-acrdn-
item')).getElementsByClassName('e-tgl-collapse-icon').length)) {
            var defaultSidebar =
ej.base.getComponent(document.getElementById("default-sidebar"), "sidebar");
            defaultSidebar.hide();
        }
    }
</script>

```

SIDEBAR.CS

```

// SideBar
public ActionResult SideBar()
{
    return View();
}

```

Mobile view

The following example demonstrates the use case of Menu in Mobile mode by using ListView component with hamburger.

CSHTML

```

<div id='container'>
  <div class="layoutWrapper">
    <div class="speaker">
      <div class="camera"></div>
    </div>
    <div class="layout">
      <div id="container">
        <div id="header">
          <span id="hamburger" class="e-icons menu
default"></span>
          <div class="content">Header</div>
        </div>
        <!-- ListView element -->

@Html.EJS().ListView("listview").DataSource(ViewBag.Items).HeaderTitle("Menu
").ShowHeader(true).Select("onSelect").Render()
          <span id='close' style="display: none" class='e-icons'
onclick="onCloseClick()"></span>
        </div>
      </div>
      <div class="outerButton"> </div>
    </div>
  </div>
</div>
<script>
  document.addEventListener("DOMContentLoaded", function () {
    var listObj =
ej.base.getComponent(document.getElementById("listview"), "listview");
    listObj.element.style.display = 'none';
  });
  document.getElementById('hamburger').onclick = function () {
    var listObj =
ej.base.getComponent(document.getElementById("listview"), "listview");
    var animation = new ej.base.Animation
      ({ duration: 500 });
    animation.animate(listObj.element, {
      name: 'SlideDown',
      begin: function (args) {
        listObj.element.style.display = 'block';
        document.getElementById('close').style.display =
'block';
      }
    });
  };
  function onSelect(e) {
    var listObj =
ej.base.getComponent(document.getElementById("listview"), "listview");
    if (e.data && !e.data.child) {
      listObj.element.style.display = 'none';
      document.getElementById('close').style.display = 'none';
      listObj.refresh();
    }
  }
  function onCloseClick() {
    var listObj =
ej.base.getComponent(document.getElementById("listview"), "listview");
    listObj.element.style.display = 'none';
    document.getElementById('close').style.display = 'none';
  }

```



```

    }
</script>
<style>
    .layoutWrapper {
        line-height: initial;
        border: 1px solid black;
        width: 285px;
        height: 505px;
        margin: auto;
        margin-bottom: 15px;
        border-radius: 28px;
        position: relative;
    }
    .layoutWrapper .speaker {
        border: 1px solid black;
        border-radius: 5px;
        width: 33.33333333%;
        height: 5px;
        margin: 15px auto 0px auto;
        position: relative;
    }
    .layoutWrapper .outerButton {
        width: 30px;
        height: 30px;
        border: 1px solid black;
        border-radius: 50%;
        position: absolute;
        bottom: calc(0% + 10px);
        left: calc(50% - 15px);
    }
    .layoutWrapper .camera {
        position: absolute;
        left: calc(-15% - 10px);
        top: -100%;
        width: 10px;
        height: 10px;
        border-radius: 50%;
        border: 1px solid black;
    }
    .layoutWrapper .layout {
        border: 1px solid black;
        margin: 20px 13px 0px 13px;
    }
    .layout #container {
        height: 405px;
    }
    #header {
        width: 100%;
        background-color: #7b8cfb;
    }
    .layout .e-listview .e-list-header,
    .layout .e-listview .e-but-back {
        background-color: #7b8cfb;
        color: white;
    }
    #header .content {
        background-color: #7b8cfb;

```

```

        color: white;
        border: none;
        font-size: 20px;
        line-height: 50px;
        text-align: center;
    }
    #hamburger.menu {
        font-size: 25px;
        cursor: pointer;
        float: left;
        line-height: 50px;
        position: absolute;
        z-index: 1000;
        left: 25px;
        color: white;
    }
    #hamburger.menu:before {
        content: '\e99a';
    }
    #listview {
        position: absolute;
        width: 254px;
        overflow: hidden;
        top: 43px;
        left: 14px;
        z-index: 1000;
    }
    #close:before {
        content: "\e945";
    }
    #close {
        position: absolute;
        right: 25px;
        top: 58px;
        z-index: 10000;
        color: white;
    }
</style>

```

LISTVIEW.CS

```

// Mobile view data
public ActionResult ListView()
{
    List<object> Items = new List<object>() {
        new {
            text = "Appliances",
            id = "list1",
            child = new List<object>() {
                new {
                    text = "Kitchen",
                    id = "list1_1",
                    child = new List<object>() {
                        new { id = "list1_1_1" , text= "Electric Cookers" },
                        new { id = "list1_1_2" , text= "Coffee Makers" },
                        new { id = "list1_1_3" , text= "Blenders" }
                    }
                }
            }
        }
    };
}

```

```

    },
    new {
        text = "Washing Machine",
        id = "list1_2",
        child = new List<object>() {
            new { id = "list1_2_1", text= "Fully Automatic" },
            new { id = "list1_2_2", text= "Semi Automatic" }
        }
    },
    new {
        text = "Air Conditioners",
        id = "list1_3",
        child = new List<object>() {
            new { id = "list1_3_1", text= "Inverter ACs" },
            new { id = "list1_3_2", text= "Split ACs" },
            new { id = "list1_3_3", text= "Window ACs" }
        }
    }
},
new {
    text = "Accessories",
    id = "list2",
    child = new List<object>() {
        new {
            text = "Mobile",
            id = "list2_1",
            child = new List<object>() {
                new { id = "list2_1_1", text= "Headphones" },
                new { id = "list2_1_2", text= "Memory Cards" },
                new { id = "list2_1_3", text= "Power Banks" }
            }
        },
        new {
            text = "Computer",
            id = "list2_2",
            child = new List<object>() {
                new { id = "list2_2_1", text= "Pendrives" },
                new { id = "list2_2_2", text= "External Hard Disks" },
                new { id = "list2_2_3", text= "Monitors" }
            }
        }
    }
},
new {
    text = "Fashion",
    id = "list3",
    child = new List<object>() {
        new { id= "list3_1", text = "Men" },
        new { id= "list3_2", text = "Women" }
    }
},
new {
    text = "Home & Living",
    id = "list4",

```

```

        child = new List<object>() {
            new { id= "list4_1", text = "Furniture" },
            new { id= "list4_2", text = "Decor" }
        },
        new {
            text = "Entertainment",
            id = "list5",
            child = new List<object>() {
                new { id= "list5_1", text = "Televisions" },
                new { id= "list5_2", text = "Home Theatres" },
                new { id= "list5_3", text = "Gaming Laptops" },
            }
        }
    };
    ViewBag.Items = Items;
    return View();
}

```

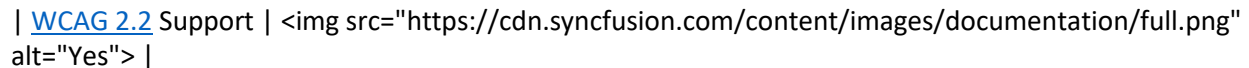
Accessibility in Menu Control

The Menu component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Menu component is outlined below.

| Accessibility Criteria | Compatibility |

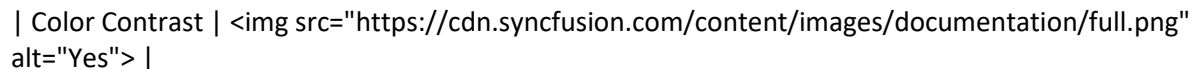
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |

| Keyboard Navigation Support |  alt="Yes" > |

| [Accessibility Checker](#) Validation |  alt="Yes" > |

| [Axe-core](#) Accessibility Validation |  alt="Yes" > |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

WAI-ARIA attributes

The Menu component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Menu component:

- | Attributes | Purpose |
|------------------|--|
| role | Indicates Menu component's root menu as <code>menubar</code> , popup as <code>menu</code> , and the popup items as <code>menuitem</code> . |
| aria-haspopup | Indicates the availability and type of interactive popup element. |
| aria-expanded | Indicates whether the subtree can be expanded or collapsed, and indicates whether its current state can be expanded or collapsed. |
| aria-orientation | Indicates whether the orientation is horizontal or vertical. The default orientation is horizontal. |
| aria-label | Indicates the menu item text. |
| aria-disabled | Indicates the state of menu item whether it is disabled. |

Keyboard interaction

The Menu component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Menu component.

- | Press | To do this |
|-------|---|
| Esc | Closes the sub menu that contains focus and returns focus to the parent element. |
| Enter | Opens the sub menu if focused menu item has sub menu, and places focus on its first item or activates the item and closes the sub menu. |
| Up | Navigates up or to the previous menu item. |
| Down | Navigates down or to the next menu item. |

| Left | Closes the current sub menu and navigates to the parent menu. |

| Right | Navigates and open the next sub menu. |

| Home | Focuses the first item. |

| End | Focuses the last item.

Ensuring accessibility

The Menu component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Menu component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Menu component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

Styles and Appearances

To modify the Menu appearance, you need to override the default CSS of Menu component. Find the list of CSS classes and its corresponding section in Menu component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

CSS Class | Purpose of Class

|.e-menu-wrapper|To customize the menu wrapper

|.e-menu-wrapper.e-menu-popup|To customize the menu popup wrapper

|.e-menu-wrapper ul .e-menu-item|To customize the menu items

|.e-menu-wrapper.e-menu-popup .e-menu-item|To customize the menu popup items

|.e-menu-wrapper ul .e-menu-item .e-caret|To customize the menu items caret icon

How To

Change orientation in Menu Control

Orientation in menu items can be changed horizontally or vertically using the [orientation](#) property. By default, it is horizontally aligned.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Orientation(ViewBag.orientation).Render()
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>
```

ORIENTATION.CS

```
using Syncfusion.EJ2.Navigations;
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
namespace EssentialJS2MvcApplication2.Controllers
{
    public class OrientationController : Controller
    {
        // GET: Orientation
        public ActionResult Orientation()
        {
            List<object> menuItems = new List<object>();
            menuItems.Add(new
            {
                text = "File",
                items = new List<object>()
                {
                    new { text = "Open" },
                    new { text = "Save" },
                    new { text = "Exit" }
                }
            });
            menuItems.Add(new
            {
                text = "Edit",
                items = new List<object>()
                {
                    new { text = "Cut" },
                    new { text = "Copy" },
                    new { text = "Paste" }
                }
            });
            menuItems.Add(new
            {
                text = "View",
                items = new List<object>()
                {
                    new { text = "Toolbar" },
                    new { text = "Sidebar" },
                    new { text = "Fullscreen" }
                }
            });
            menuItems.Add(new
            {
                text = "Tools",
                items = new List<object>()
                {
                    new { text = "Spelling & Grammar" },
                    new { text = "Customize" },
                    new { text = "Options" }
                }
            });
            menuItems.Add(new
            {
                text = "Help"
            });
        }
    }
}
```

```

        ViewBag.orientation =
Syncfusion.EJ2.Navigations.Orientation.Vertical;
        ViewBag.menuItems = menuItems;
        return View();
    }
}
}

```

Customize menu using events

The Menu provides a set of **events** to enable users to customize it.

CSHTML

```

<div class="control-section">
    <div class="menu-section">

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeOpen("updateEventLog")
.BeforeClose("updateEventLog").OnClose("updateEventLog").OnOpen("updateEventLog")
.Select("updateEventLog").Render()
    </div>
    <div class="property-section">
        <table id="propertyTable" title="Event trace">
            <tbody>
                <th>Event trace:-</th>
                <tr>
                    <td></td>
                </tr>
            </tbody>
        </table>
    </div>

@Html.EJS().Button("clear").Content("Clear").Created("onClearBtnCreated").Render()
</div>
<style>
    html, body, .control-section {
        height: 95%;
    }
    .menu-section {
        margin-top: 100px;
        text-align: center;
        float: left;
    }
    .property-section {
        overflow: auto;
        width: 40%;
        height: 330px;
        float: right;
        font-family: monospace;
    }
    .property-section th {
        text-align: left;
    }
    #clear {
        float: right;
        clear: both;
    }

```



```

    }
</style>
<script>
    function updateEventLog(args) {
        var propertyElem = document.getElementById('propertyTable');

        propertyElem.getElementsByTagName('td')[0].insertAdjacentHTML('beforeend',
        args.name + ' Event triggered. <br />');
    }
    function onClearBtnCreated() {
        document.getElementById("clear").onclick = function () {
            var propertyElem = document.getElementById('propertyTable');
            propertyElem.getElementsByTagName('td')[0].innerHTML = '';
        };
    }
</script>

```

HANDLEEVENTS.CS

```

// GET: HandleEvents
public ActionResult HandleEvents()
{
    List<object> menuItems = new List<object>() {
        new {
            text = "Events",
            items = new List<object>() {
                new { text= "Conferences" },
                new { text= "Music" },
                new { text= "Workshops" }
            }
        },
        new {
            text = "Movies",
            items = new List<object>() {
                new { text= "Now Showing" },
                new { text= "Coming Soon" }
            }
        },
        new {
            text = "Directory",
            items = new List<object>() {
                new { text= "Media Gallery" },
                new { text= "Newsletters" }
            }
        },
        new {
            text = "Queries",
            items = new List<object>() {
                new { text= "Our Policy" },
                new { text= "Site Map"},
                new { text= "24x7 Support"}
            }
        },
        new { text= "Services" }
    };

    ViewBag.menuItems = menuItems;
}

```

```
return View();
}
```

Customize menu items

Add or remove menu items

Menu items can be added or removed using the `insertAfter`, `insertBefore` and `removeItems` methods.

In the following example, the **Europe** menu items are added before the **Oceania** item, the **Africa** menu items are added after the **Asia**, and the **South America** and **Mexico** items are removed from menu.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Fields(ViewBag.menuFields)
.Created("created").Render()
<script>
function created(args) {
    var insertBefore = [
        {
            continent: 'Europe',
            countries: [
                { country: 'Finland' },
                { country: 'Austria' }
            ]
        }
    ];
    var insertItem = [
        {
            continent: 'Africa',
            countries: [
                { country: 'Nigeria' }
            ]
        }
    ];
    menuObj = document.getElementById("menu").ej2_instances[0];
    // Add items before to 'Oceania'
    menuObj.insertBefore(insertBefore, "Oceania", false);
    // Add items after to 'Asia'
    menuObj.insertAfter(insertItem, "Asia", false);
    // Remove items
    menuObj.removeItems(['South America', 'Mexico'], false);
}
</script>
<style>
body {
    margin-top: 100px;
    text-align: center;
}
</style>
```

CUSTOMIZEMENUITEMS.CS

```
// Customize Menu Items
public ActionResult CustomizeMenuItems()
{
    List<object> menuItems = new List<object>();
```

```

menuItems.Add(new
{
    continent = "Asia",
    countries = new List<object>()
    {
        new { country = "China" },
        new { country = "India" },
        new { country = "Japan" }
    },
});
menuItems.Add(new
{
    continent = "North America",
    countries = new List<object>()
    {
        new { country = "Canada" },
        new { country = "Mexico" },
        new { country = "USA" }
    },
});
menuItems.Add(new
{
    continent = "South America",
    countries = new List<object>()
    {
        new { country = "Brazil" },
        new { country = "Colombia" },
        new { country = "Argentina" }
    },
});
menuItems.Add(new
{
    continent = "Oceania",
    countries = new List<object>()
    {
        new { country = "Australia" },
        new { country = "New Zealand" },
        new { country = "Samoa" }
    },
});
menuItems.Add(new { continent = "Antarctica" });
MenuFieldSettings menuFields = new MenuFieldSettings()
{
    Text = new string[] { "continent", "country" },
    Children = new string[] { "countries" }
};
ViewBag.menuItems = menuItems;
ViewBag.menuFields = menuFields;
return View();
}

```

Note: To process items with ID values, set `isUnique` to `true`.

Enable or disable menu items

You can enable and disable the menu items using the `enableItems` method in Menu. To enable menu items, set the `enable` property in argument to `true` and vice-versa.

In the following example, the **Directory** header item, **Conferences**, and **Music** sub menu items are disabled.

CSHTML

```
@Html.EJS().Button("enableAll").Content("Enable all
items").Created("btnCreated").Render()
<div class="menu-section">

@Html.EJS().Menu("menu").Items(ViewBag.data).Created("created").BeforeOpen("
beforeOpen").Render()
</div>
<style>
    .menu-section {
        margin-top: 100px;
        text-align: center;
    }
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>
<script>
    var disableItems = ['Conferences', 'Music', 'Directory'], menuObj;
    function beforeOpen(args) {
        menuObj = ej.base.getComponent(document.getElementById("menu"),
"menu");
        //Handling sub menu items
        for (var i = 0; i < args.items.length; i++) {
            if (disableItems.indexOf(args.items[i].text) > -1) {
                menuObj.enableItems([args.items[i].text], false, false);
            }
        }
    }
    function created(args) {
        menuObj = ej.base.getComponent(document.getElementById("menu"),
"menu");
        //Disable items
        menuObj.enableItems(disableItems, false, false);
    }
    function btnCreated(args) {
        var buttonObj =
ej.base.getInstance(document.getElementById("showAll"), ej.buttons.Button);
        buttonObj.element.onclick = () => {
            //Enable items
            menuObj.enableItems(disableItems, true, false);
            disableItems = [];
        }
    }
</script>
```

ENABLEDISABLE.CS

```
// Enable and Disable Menu Items
public ActionResult EnableDisable()
{
    List<object> data = new List<object>() {
        new {
            text = "Events",
            items = new List<object>() {
                new { text= "Conferences" },
                new { text= "Music" },
                new { text= "Workshops" }
            }
        },
        new {
            text = "Movies",
            items = new List<object>() {
                new { text= "Now Showing" },
                new { text= "Coming Soon" }
            }
        },
        new {
            text = "Directory",
            items = new List<object>() {
                new { text= "Media Gallery" },
                new { text= "Newsletters" }
            }
        },
        new {
            text = "Queries",
            items = new List<object>() {
                new { text= "Our Policy" },
                new { text= "Site Map" },
                new { text= "24x7 Support" }
            }
        },
        new { text= "Services" }
    };
    ViewBag.data = data;
    return View();
}
```

Note: To disable sub menu items, use the [beforeOpen](#) event.

Hide or show menu items

You can show or hide the menu items using the `showItems` and `hideItems` methods.

In the following example, the **Movies** header item, **Workshops**, and **Music** sub menu items are hidden in menu.

CSHTML

```
@Html.EJS().Button("showAll").Content("Show All
items").Created("btnCreated").Render()
<div class="menu-section">
```

```

@Html.EJS().Menu("menu").Items(ViewBag.data).Created("created").BeforeOpen("
beforeOpen").Render()
</div>
<style>
    .menu-section {
        margin-top: 100px;
        text-align: center;
    }
    body {
        margin-top: 100px;
        text-align: center;
    }
</style>
<script>
    var hiddenItems = ['Workshops', 'Music', 'Movies'], menuObj;
    function beforeOpen(args) {
        menuObj = ej.base.getComponent(document.getElementById("menu"),
"menu");
        //Handling sub menu items
        for (var i = 0; i < args.items.length; i++) {
            if (hiddenItems.indexOf(args.items[i].text) > -1) {
                menuObj.hideItems([args.items[i].text], false);
            }
        }
    }
    function created(args) {
        menuObj = ej.base.getComponent(document.getElementById("menu"),
"menu");
        menuObj.hideItems(hiddenItems, false);
    }
    function btnCreated(args) {
        var buttonObj =
ej.base.getInstance(document.getElementById("showAll"), ej.buttons.Button);
        buttonObj.element.onclick = () => {
            menuObj.showItems(hiddenItems, false);
            hiddenItems = [];
        }
    }
</script>

```

HIDESHOW.CS

```

public ActionResult ShowHide()
{
    List<object> data = new List<object>() {
        new {
            text = "Events",
            items = new List<object>() {
                new { text= "Conferences" },
                new { text= "Music" },
                new { text= "Workshops" }
            }
        },
        new {
            text = "Movies",

```

```

        items = new List<object>() {
            new { text= "Now Showing" },
            new { text= "Coming Soon" }
        },
        new {
            text = "Directory",
            items = new List<object>() {
                new { text= "Media Gallery" },
                new { text= "Newsletters" }
            }
        },
        new {
            text = "Queries",
            items = new List<object>() {
                new { text= "Our Policy" },
                new { text= "Site Map"},
                new { text= "24x7 Support"}
            }
        },
        new { text= "Services" }
    };
    ViewBag.data = data;
    return View();
}

```

Note: Using the [beforeOpen](#) event, you can hide the sub menu items as in the above example since the menu supports to hide items only for headers initially.

Create mnemonic UI in menu item

A particular character in a text can be underlined in the [beforeItemRender](#) event by adding `<u>` tag in between the text and assign the innerHTML to the `li` element.

In the following example, the first character in `File`, `Open`, and `Save` menu items are underlined.

In the below example, `File`, `Open` and `Save` items are underlined with first character in menu items.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeItemRender("beforeItemRender").Render()
<script>
    function beforeItemRender(args) {
        if (['File', 'Open', 'Save'].indexOf(args.item.text) > -1) {
            // To underline a First character.
            var underlinedText = '<u>' + args.item.text.slice(0, 1) + '</u>'
            + args.item.text.slice(1);
            args.element.innerHTML =
            args.element.innerHTML.replace(args.item.text, underlinedText);
        }
    }
</script>
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }

```

```

}
/**
 * ej2 Menu styles
 */
@@font-face {
  font-family: 'e-menu';
  src:
    url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjvJSpgAAAEoAAAAVmNtYXBsm2feAAABpAAAAGxnbHlmmEc
yrQAAAIQAAAWIaGVhZBj0bwcAAADQAAAAANmhoZWEHmQNyAAAArAAAACRobXR4I0AAAAAAAYAAAAA
kbG9jYQaGB+4AAAIQAAAFglheHABGACaAAABCAAAACBuYW1lc0cOBgAAB6wAAAIlcG9zdJbKd4k
AAAnUAAAFQABAAADUv9qAFoEAAAA//4D6gABAAAAAAAAAAAAAAAAAAAAACQABAAAAQAahka7o18
PPPUACwPoAAAAANe2FRwAAAAA17YVHAAAAAAD6gPqAAAAACAACAAAAAAAAAAAAAAAJAI4ABQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAAQPrAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5anohgNS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAAAAAAAgAAAAAMAAAAUAMAAQAAABQABABYAAAAADgA
IAAIABuWp5bPluefo6CLohv//AADlqeWy5bnn6Ogi6IX//wAAAAAAAAAAAAAAAAABAA4ADgAQABA
AEAAQAAAABQAGAAcACAABAAIAAwAEAAAAAACsASoBRAGwAhICUAKEAsQABAAAAAAD6gNZAD8AfwC
DAI0AAAEzHw0dAQ8OLw8/DiMzHw0dAQ8OLw8/DgMhAyEBIRU3EyUVBQMjAwgJCAgIBwcHBgUFBAQ
DAgEBAgMEBAUFBgCHBwgICAKJCAgIBwcHBgUFBAQDAgEBAQECAwQEBQUGBwcHCAGI5AgJCAgHBwc
GBQUEBAIDAQEDAgQEBQUGBwcHCAGJCAKICAgIBWYGBQUFAwMCAQEBAQIDAuUFBQYGBwgICAIjAny
Q/qj+EgEKAssBcf5Yy9UBTWICAgQEBQYGBgcHCAGJCAKICQcIBWYGBgQFAwMCAQEBAQIDAUEBgY
GBwgHCQgJCAKICACHBgYGBQQEAgICAgICBAQFBgYGBwcICAKICQgJBwgHBgYGBAUDAwIBAQEBAgM
DBQQGBgYHCACJCAKICQgIBwcGBgYFBAQCAgIBu/67AZUBaf5LAj0CABUAAAAFAAAAAAPqA+oAAgA
WABgAPABkAAA3QEnMx8PFQc3MQE7AR8OAQcvDwEzHw0PBi8PPwP/nAgODg4NDawLCwoICAcFBAM
C6k4CdAgHEA4PDQ0MDA0JCACGBAIB/kWFAQMEBgJCgSLDQ0NDg4ODgLaBg0GBgYGBjwFBAMBAQE
CAGYJNAEDBAYHCQoKDAWNDQ4ODg40GQkKZJsBAwQFBwcJCQoMCw0NDg8OCE7pAnUDBQYHCQkLDAW
NDg0ODg7+SIgODg4NDg0MDAsKCAgGBAMBARUCAgMDBQU9CQkJCQgICAcNDjQNDg4ODQ0MDAsJCQc
GBAMBNA4DagAAAAABAAAAAAPqA60ACgAAEYEVIRUhAxMhAyEVAcwBzPzEN5MDHrj84gOtXfz9/QG
n/boAAAAABQAAAAADjgPqAAMABwALAA8AUwAAEYEVITUhfSE1IRUhJxEhESUhHw8RDw8hLw8RPw7
qAij92AIo/dgCKP3YOWKi/XICegGICAgHBWYGBgQUEAwMCAQEBAQIDAuUFBQYGBwcICAgI/YYICAg
IBwcGBgYFBAMDAGEBACAwMEBQUGBgCHCAGIAQs+9j72Prj9XgKi9gEBAgMDBAUFBgYHBwgICAJ
8zgGICAgHBWYGBgQUEAwMCAQEBAQIDAuUFBQYGBwcICAgIAzIICAgIBwcGBgYFBAMDAGEBQAAAAA
DqQOpAAQACgAUAB4A0wAACQEXATUBFAcmNDIDBgcuATQ2MhYUAWYHLgE0NjIWFBC2NS4BIgYUfhc
yNxCHJiMOARQWmjY3NCc3ATM1Ayb++FkBMv5fFRUq3xglJjExSzEZGCUmMTFLMUoOAmKUY2JLJyF
mZiEnS2JilWICDmCBM4MDgP74WQE2K/50FQICKv6lGQICMkoyMkoB9xkCAjJKmjJKIyEnSmNj1GM
CDmdnDgJj1GNjSichZ/7NKwAAAAAMAAAAA4oD5gAHABAAJwAAARUhNTMRIRElHgEGiIy0NjInBgC
jIgYVERQWMyEyNjURNcyRAs4BIgEZAbzd/ZABWAwBgIYZGSZhIg+8JjU1JgJ2JjU1JrwPRFGdLn5
9/TICz1IMJxkZJxLAGSkzJv0pJzMzJwLXJjMpMwADAAAAAOpA+cAAwAUAB4AAAERIREnBhURFBY
XIT4BNRE0JiMhIicGFREzESE1IQYDTP4MQxs2JgH3JzU1J/4JJtgZXQIT/egmAs/9jwJxRBkm/Yc
mMwICMyYCeSYnXon/Y8CcV4CAIAAAAAA+cD5wALACMAAAEOAQcuASc+ATceAQUeARcyNj8BARY
yNjQnATc+ATUuAScOaQLYA7SHiLMEBLOiH7T9KwXnrkeBNAMBAQ4kHA7+/wMpLgTora7nAk6HtAM
DtIeIswQEs4it6AQUKQP+/w4bJQ4BAQM0gUeu5wUf5wAAAAASAN4AAQAAAAAAAAABAAAAQAAAAA
AAQAHAAEAAQAAAAAAAAgAHAAGAAQAAAAAAAAAwAHAA8AAQAAAAAAAAABAHABYAAQAAAAAAAAABQALAB0AAQA
AAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBAkAAAAACAG0AAwABBAkAAQAOAG8
AAwABBAkAAgAOAH0AAwABBAkAAwAOAISAAwABBAkABAAOAJkAAwABBAkABQAWAKcAAwABBAkABgA
OAL0AAwABBAkACgBYAMsAAwABBAkACwAkASMgZS1pY29uc1JlZ3VsYXJlLW1jb25zZS1pY29uc1Z
1cnNpb24gMS4wZS1pY29uc0ZvbnQgZ2ZuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R
lZG1vd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwBSAGUAZwB1AGwAYQByAGUALQB
pAGMABwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkABwBuACAAMQAuADAAZQAtAGkAYwBvAG4
AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQAIAb1AHMAAQBuAGcAIABTAHkAbgBjAGYAdQB
zAGkABwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGk
ABwBuAC4AYwBvAG0AAAAAAGAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJAQIBAwEEAQU
BBgEHAQgBCQEKAAlzaG9wcGluZy1jYXJ0B2VkaXQtMDUMZmlsZS1vcGVuLTAxZGZpbGUtdGV4dC0
wMQNDdXQFUGFzdGUEQ29weQZTZWFyY2gAAAAA==) format('trueType');
  font-weight: normal;
  font-style: normal;

```



```

}
.em-icons {
    font-family: 'e-menu';
    font-style: normal;
    font-variant: normal;
    font-weight: normal;
    line-height: 1;
    text-transform: none;
}

.e-file::before {
    content: '\e886';
}

.e-edit::before {
    content: '\e822';
}

.e-cut::before {
    content: '\e5a9';
}

.e-copy::before {
    content: '\e5b3';
}

.e-paste::before {
    content: '\e5b2';
}

.e-open::before {
    content: '\e885';
}

.e-save::before {
    content: '\e98e';
}
</style>

```

UNDERLINE.CS

```

// Underline
public ActionResult Underline()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "File",
        iconCss = "em-icons e-file",
        items = new List<object>()
        {
            new { text = "Open", iconCss= "em-icons e-open" },
            new { text = "Save", iconCss= "e-icons e-save" },
            new { separator = true },
            new { text = "Exit" }
        }
    });
    menuItems.Add(new
    {
        text = "Edit",
        iconCss = "em-icons e-edit",

```

```

        items = new List<object>()
        {
            new { text = "Cut", iconCss= "em-icons e-cut" },
            new { text = "Copy", iconCss= "em-icons e-copy" },
            new { text = "Paste", iconCss= "em-icons e-paste" }
        }
    });
    menuItems.Add(new
    {
        text = "Format",
    });
    menuItems.Add(new
    {
        text = "View"
    });
    menuItems.Add(new
    {
        text = "Bookmarks"
    });
    menuItems.Add(new
    {
        text = "Tools"
    });
    menuItems.Add(new
    {
        separator = true
    });
    menuItems.Add(new
    {
        text = "Help"
    });
    ViewBag.menuItems = menuItems;
    return View();
}

```

Change sub menu position

The submenu position can be changed by using the [beforeOpen](#) event. Assign the top and left position where you want to open the submenu to the [beforeOpen](#) event arguments `args.top` and `args.left` respectively.

In the below sample, the sub menu opens above the parent menu item.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeOpen('onBeforeOpen')
.Render()
<script>
    function onBeforeOpen(args) {
        // Getting parent menu item element offset
        var relativeOffset = ej.base.closest(args.event.target, '.e-menu-
item').getBoundingClientRect();
        // Getting sub menu wrapper element using closest method
        var subMenuEle = ej.base.closest(args.element, '.e-menu-wrapper');
        subMenuEle.style.display = 'block';
        args.top = (relativeOffset.top -
subMenuEle.getBoundingClientRect().height) + pageYOffset;
    }

```

```

        args.left = relativeOffset.left + pageXOffset;
        subMenuEle.style.display = '';
    }
</script>
<style>
    body {
        margin-top: 200px;
        text-align: center;
    }
</style>

```

POSITION.CS

```

public ActionResult Position()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "File",
        items = new List<object>()
        {
            new { text = "Open" },
            new { text = "Save" },
            new { text = "Exit" }
        }
    });
    menuItems.Add(new
    {
        text = "Edit",
        items = new List<object>()
        {
            new { text = "Cut" },
            new { text = "Copy" },
            new { text = "Paste" }
        }
    });
    menuItems.Add(new
    {
        text = "View",
        items = new List<object>()
        {
            new { text = "Toolbar" },
            new { text = "Sidebar" },
            new { text = "Fullscreen" }
        }
    });
    menuItems.Add(new
    {
        text = "Tools",
        items = new List<object>()
        {
            new { text = "Spelling & Grammar" },
            new { text = "Customize" },
            new { text = "Options" }
        }
    });
}

```

```

menuItem.Add(new
{
    text = "Help"
});
ViewBag.menuItems = menuItem;
return View();
}

```

Note: For custom positioning, set both **top** and **left** position in the [beforeOpen](#) event.

Menu with rounded corner

The rounded corner can be achieved by using the [cssClass](#) property. Add a custom class to the menu component and customize it using the **border-radius** CSS property. For more information, refer to the styles specified in the below sample.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).CssClass('e-rounded-menu').showItemOnClick("true").Render()
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
    /* Styles to achieve rounder corner in menu */
    .e-menu-wrapper.e-rounded-menu:not(.e-menu-popup),
    .e-menu-wrapper.e-rounded-menu .e-menu {
        border-radius: 20px;
    }
    /* Increased the menu component left and right padding for better
    rounded corner UI */
    .e-menu-wrapper.e-rounded-menu ul.e-menu {
        padding: 0 14px;
    }
</style>

```

ROUNDED.CS

```

public ActionResult Rounded()
{
    List<object> menuItem = new List<object>();
    menuItem.Add(new
    {
        text = "File",
        items = new List<object>()
        {
            new { text = "Open" },
            new { text = "Save" },
            new { text = "Exit" }
        }
    });
    menuItem.Add(new
    {
        text = "Edit",
        items = new List<object>()
    });
}

```

```

        {
            new
            {
                text= "Toolbars",
                items = new List<object>()
                {
                    new { text= "Menu Bar"},
                    new { text= "Bookmarks Toolbar"}
                }
            },
            new
            {
                text = "Zoom",
                items =new List<object>()
                {
                    new { text= "Zoom In"},
                    new { text= "Zoom Out"},
                    new { text= "Reset"}
                }
            },
            new
            {
                text = "Full Screen",
                items =new List<object>()
                {
                    new { text= "cancel"}
                }
            },
        }
    });
    menuItems.Add(new
    {
        text = "View",
        items = new List<object>()
        {
            new { text = "Toolbar" },
            new { text = "Sidebar" },
            new { text = "Fullscreen" }
        }
    });
    menuItems.Add(new
    {
        text = "Tools",
        items = new List<object>()
        {
            new { text = "Spelling & Grammar" },
            new { text = "Customize" },
            new { text = "Options" }
        }
    });
    menuItems.Add(new
    {
        text = "Help"
    });
    ViewBag.menuItems = menuItems;
    return View();
}

```

Set title for Menu Items

In this sample, the title for settings icon can be achievable by using `beforeItemRender` client-side event in Menu component.

CSHTML

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeItemRender("itemRender").Render()
<script>
function itemRender(args) {
    if (args.item.id == 'settingIcon') {
        args.element.setAttribute('title', 'Settings');
    }
}
</script>
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
    .em-icons {
        font-family: 'e-menu';
        font-style: normal;
        font-variant: normal;
        font-weight: normal;
        line-height: 1;
        text-transform: none;
    }
    .e-file::before {
        content: '\e886';
    }
    .e-menu-wrapper .e-menu .e-menu-item.e-menu-caret-icon .e-icons.e-caret {
        display: none;
    }

    .e-menu-wrapper .e-menu .e-menu-item.e-menu-caret-icon {
        padding-right: 5px;
    }
</style>
```

TITLE.CS

```
public ActionResult Title()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        id: 'settingIcon',
        iconCss: 'em-icons e-file',
        items = new List<object>()
        {
            new { text = "Open",
                items = new List<object>()
            }
        }
    });
}
```

```

        new { text = "Option 1" },
        new { text = "Option 2" }
    } },
    new { text = "Save" },
    { separator: true },
    new { text = "Exit" }
}
});
ViewBag.menuItems = menuItems;
return View();
}

```

Open menu and sub menu on click only

You can open menu items and sub menu on menu item click by setting [showItemOnClick](#) property of the Menu. To open sub menu items only on item click, should be set as `true`.

CSHTML

```

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).CssClass('e-rounded-menu').showItemOnClick("true").Render()
<style>
    body {
        margin-top: 100px;
        text-align: center;
    }
    /* Styles to achieve rounder corner in menu */
    .e-menu-wrapper.e-rounded-menu:not(.e-menu-popup),
    .e-menu-wrapper.e-rounded-menu .e-menu {
        border-radius: 20px;
    }
    /* Increased the menu component left and right padding for better rounded corner UI */
    .e-menu-wrapper.e-rounded-menu ul.e-menu {
        padding: 0 14px;
    }
</style>

```

ROUNDED.CS

```

public ActionResult Rounded()
{
    List<object> menuItems = new List<object>();
    menuItems.Add(new
    {
        text = "File",
        items = new List<object>()
        {
            new { text = "Open" },
            new { text = "Save" },
            new { text = "Exit" }
        }
    });
    menuItems.Add(new
    {
        text = "Edit",

```

```

        items = new List<object>()
        {
            new
            {
                text= "Toolbars",
                items = new List<object>()
                {
                    new { text= "Menu Bar"},
                    new { text= "Bookmarks Toolbar"}
                }
            },
            new
            {
                text = "Zoom",
                items =new List<object>()
                {
                    new { text= "Zoom In"},
                    new { text= "Zoom Out"},
                    new { text= "Reset"}
                }
            },
            new
            {
                text = "Full Screen",
                items =new List<object>()
                {
                    new { text= "cancel"}
                }
            },
        }
    });
    menuItems.Add(new
    {
        text = "View",
        items = new List<object>()
        {
            new { text = "Toolbar" },
            new { text = "Sidebar" },
            new { text = "Fullscreen" }
        }
    });
    menuItems.Add(new
    {
        text = "Tools",
        items = new List<object>()
        {
            new { text = "Spelling & Grammar" },
            new { text = "Customize" },
            new { text = "Options" }
        }
    });
    menuItems.Add(new
    {
        text = "Help"
    });
    ViewBag.menuItems = menuItems;
    return View();

```


| |
|---|
| } |
|---|

Migration from Essential JS 1

This article describes the API migration process of Menu component from Essential JS 1 to Essential JS 2.

Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Animation type on hover or click on the menu items | **Property:** *animationType*

@Html.EJ().Menu("menu").AnimationType(AnimationType.Default) | Not applicable |

| Context menu target | **Property:** *contextMenuTarget*

@Html.EJ().Menu("menu").ContextMenuTarget("#target") | Not applicable |

| Container element for submenu's collision | **Property:** *container*

@Html.EJ().Menu("menu").ContextMenuTarget("#target").Container("#target") | Not applicable |

| Menu items | Not applicable | **Property:** *items*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render() |

| Properties of menu items | **Property:** *id, text, imageUrl, children and url*

@Html.EJ().Menu("menu").Items(items =>
 {

items.Add().Id("fileid").Text("File");

items.Add().Id("editid").Text("Edit").Children(child =>
 {

child.Add().Text("Cut");
 child.Add().Text("Copy");

child.Add().Text("Paste");
 });

items.Add().Id("viewid").Text("View").Url("#");

items.Add().Id("helpid").Text("Help");
 }) | **Property:** *id, text, items, iconCss and url*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()
 List<object> menuItems =
new List<object>();
menuItems.Add(new
{
 text = "File", id =
"fileid"
});
menuItems.Add(new
{
 text = "Edit",
id= "editid",

items = new List<object>()
 {
 new { text= "Cut", iconCss= "e-icons e-
cut" },
 new { text= "Copy", iconCss= "e-icons e-copy" },
 new {
text= "Paste", iconCss= "e-icons e-paste" }
 };
menuItems.Add(new
{

 text = "view", id= "viewid", url= "#"
});
menuItems.Add(new
{
 text =
"Help", id= "helpid"
});
 ViewBag.menuItems = menuItems; |

| Adding custom class | **Property:** *cssClass*

@Html.EJ().Menu("menu").CssClass("custom-class") | **Property:** *cssClass*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).CssClass("custom-class").Render() |

| Enables or disables the animation on hover or click on the menu items | **Property:** *enableAnimation*

 @Html.EJ().Menu("menu").EnableAnimation(true) | Not applicable |

| Animation settings | Not applicable | **Property:** *animationSettings*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).AnimationSettings("ViewBag.animation").Render() |

| Root menu items to be aligned center in horizontal menu | **Property:** *enableCenterAlign* @Html.EJ().Menu("menu").EnableCenterAlign(true) | Not applicable |

| Disabled state | **Property:** *enabled* @Html.EJ().Menu("menu").Enabled(false) | **Property:** *disabled* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Disabled(true).Render() |

| RTL | **Property:** *enableRTL* @Html.EJ().Menu("menu").EnableRTL(true) | **Property:** *enableRtl* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).EnableRtl(true).Render() |

| Enables/Disables the separator | **Property:** *enableSeparator* @Html.EJ().Menu("menu").EnableSeparator(false) | Not applicable |

| Menu Type | **Property:** *menuType* @Html.EJ().Menu("menu").MenuType(MenuType.ContextMenu) | Not applicable |

| Exclude target for context menu | **Property:** *excludeTarget* @Html.EJ().Menu("menu").MenuType(MenuType.ContextMenu).ContextMenuTarget("#target").ExcludeTarget(".inner") | Not applicable |

| Fields | **Property:** *fields* @Html.EJ().Menu("menu").MenuFields(f => f.Datasource((IEnumerable<Check.Controllers.CheckController.MenuJson>)ViewBag.datasource).Id("id").Text("text").ParentId("parent")) | **Property:** *fields* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Fields("ViewBag.menuFields").Render() |

| Height | **Property:** *height* @Html.EJ().Menu("menu").Height("50") | Not applicable |

| Width | **Property:** *width* @Html.EJ().Menu("menu").Width("800") | Not applicable |

| HTML Attributes | **Property:** *htmlAttributes* @Html.EJ().Menu("menu").HtmlAttribute("") | Not applicable |

| Responsive | **Property:** *isResponsive* @Html.EJ().Menu("menu").IsResponsive(true) | Not applicable |

| Show item on click | **Property:** *openOnClick* @Html.EJ().Menu("menu").OpenOnClick(true) | **Property:** *showItemOnClick* @Html.EJS().Menu("menu").ShowItemOnClick(true).Items("ViewBag.menuItems").Render() |

| Orientation | **Property:** *orientation* @Html.EJ().Menu("menu").Orientation(Orientation.Vertical) | **Property:** *orientation* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Orientation("Vertical").Render() |

| Show root level arrows | **Property:** *showRootLevelArrows* @Html.EJ().Menu("menu").ShowRootLevelArrows(false) | Not applicable |

| Show sub level arrows | **Property:** *showSubLevelArrows* @Html.EJ().Menu("menu").ShowSubLevelArrows(false) | Not applicable |

| Sub menu direction | **Property:** *subMenuDirection* @Html.EJ().Menu("menu").SubMenuDirection(Direction.Left) | Not applicable |

| Title | **Property:** *titleText* @Html.EJ().Menu("menu").TitleText("Title of the Menu") |
Not applicable |

| Template | Not applicable | **Property:** *template*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Template("#menuTemplate").Render()
|

| Pop up menu height | **Property:** *overflowHeight*

@Html.EJ().Menu("menu").OverflowHeight("200") | Not applicable |

Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Disable Method | **Method:** *disable*

 @Html.EJ().Menu("menu")
 var menu =
\$("#menu").data("ejMenu");
 menu.disable(); | Not applicable |

| Disable menu items | **Method:** *disableItem*

 @Html.EJ().Menu("menu").Items(items =>

 {
 items.Add().Text("File");
 items.Add().Text("Edit");

items.Add().Text("Help");
 })
 var menu = \$("#menu").data("ejMenu");

menu.disableItem("File"); | **Method:** *enableItems*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()
 var menu =
document.getElementById("menu").ej2_instances[0];
 menu.enableItems("File", false) |

| Disable menu items by ID | **Method:** *disableItemByID*

@Html.EJ().Menu("menu").Items(items =>
 {

items.Add().Id("fileid").Text("File");
 items.Add().Id("editid").Text("Edit");

 items.Add().Id("helpid").Text("Help");
 })
 var menu =
\$("#menu").data("ejMenu");
 menu.disableItemByID("fileid"); | **Method:** *enableItems*

 @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()
 var menu =
document.getElementById("menu").ej2_instances[0];
 menu.enableItems("fileid", false, true) |

| Enable Method | **Method:** *enable*

 @Html.EJ().Menu("menu")
 var menu =
\$("#menu").data("ejMenu");
 menu.enable(); | Not applicable |

| Enable menu items | **Method:** *enableItem*

 @Html.EJ().Menu("menu").Items(items =>

 {
 items.Add().Text("File");
 items.Add().Text("Edit");

items.Add().Text("Help");
 })
 var menu = \$("#menu").data("ejMenu");

menu.enableItem("File"); | **Method:** *enableItems*

@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()
 var menu =
document.getElementById("menu").ej2_instances[0];
 menu.enableItems("File", true); |

| Enable menu items by ID | **Method:** *enableItemByID*

@Html.EJ().Menu("menu").Items(items =>
 {

items.Add().Id("fileid").Text("File");
 items.Add().Id("editid").Text("Edit");

 items.Add().Id("helpid").Text("Help");
 })
 var menu =
\$("#menu").data("ejMenu");
 menu.enableItemByID("fileid"); | **Method:** *enableItems*

 @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render()
 var menu =
document.getElementById("menu").ej2_instances[0];
 menu.enableItems("fileid", true, true); |

| Hide Method | **Method:** *hide* @Html.EJ().Menu("menu") var menu = \$("#menu").data("ejMenu"); menu.hide(); | Not applicable |

| Hide menu items | **Method:** *hideItems* @Html.EJ().Menu("menu").Items(items => { items.Add().Id("fileid").Text("File"); items.Add().Id("editid").Text("Edit"); items.Add().Id("helpid").Text("Help"); }) var menu = \$("#menu").data("ejMenu"); menu.hideItems(["#helpid", "#editid"]); | **Method:** *hideItems* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render() var menu = document.getElementById("menu").ej2instances[0]; menu.hideItems(["editid", "helpid"], true); |

| Insert menu items | **Method:** *insert* @Html.EJ().Menu("menu").Items(items => { items.Add().Id("fileid").Text("File"); items.Add().Id("editid").Text("Edit"); items.Add().Id("helpid").Text("Help"); }) var menu = \$("#menu").data("ejMenu"); menu.insert({ id: "viewid", text: "View" }, "#edit_id"); | Not applicable |

| Insert menu items after the specified menu item | **Method:** *insertAfter* @Html.EJ().Menu("menu").Items(items => { items.Add().Id("fileid").Text("File"); items.Add().Id("editid").Text("Edit"); items.Add().Id("helpid").Text("Help"); }) var menu = \$("#menu").data("ejMenu"); menu.insertAfter({ id: "viewid", text: "View" }, "#editid"); | **Method:** *insertAfter* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render() var menu = document.getElementById("menu").ej2instances[0]; menu.insertAfter({ id: "view_id", text: "View" }, "Edit"); |

| Insert menu items before the specified menu item | **Method:** *insertBefore* @Html.EJ().Menu("menu").Items(items => { items.Add().Id("fileid").Text("File"); items.Add().Id("editid").Text("Edit"); items.Add().Id("helpid").Text("Help"); }) var menu = \$("#menu").data("ejMenu"); menu.insertBefore({ id: "viewid", text: "View" }, "#helpid"); | **Method:** *insertBefore* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render() var menu = document.getElementById("menu").ej2instances[0]; menu.insertBefore({ id: "view_id", text: "View" }, "Help"); |

| Remove menu items | **Method:** *remove* @Html.EJ().Menu("menu").Items(items => { items.Add().Id("fileid").Text("File"); items.Add().Id("editid").Text("Edit"); items.Add().Id("helpid").Text("Help"); }) var menu = \$("#menu").data("ejMenu"); menu.remove(["#Edit"]); | **Method:** *removeItems* @Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render() var menu = document.getElementById("menu").ej2instances[0]; menu.removeItems(["Edit"]); |

| To show menu | **Method:** *show* @Html.EJ().Menu("menu") var menu = \$("#menu").data("ejMenu"); menu.show(); | Not applicable |

| Destroy method | **Method:** *destroy*

 @Html.EJ().Menu("menu")
 var menu = \$("#menu").data("ejMenu");
 menu.destroy(); | **Method:** *destroy*


```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Render() <br/> var menu =
document.getElementById("menu").ej2_instances[0]; <br/> menu.destroy(); |
```

Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers before opening the menu | **Events:** *beforeOpen*


```
@Html.EJ().Menu("menu").BeforeOpen("beforeOpen") <br/> function beforeOpen(args) { <br/>
&#160; &#160; / code block / <br/> } | Events: beforeOpen <br/><br/>
```

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeOpen("beforeOpen").Render() <br/>
function beforeOpen(args) { <br/> &#160; &#160; / code block / <br/> } |
```

| Triggers before closing the menu | Not applicable | **Events:** *beforeClose*


```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeClose("beforeClose").Render()
<br/> function beforeClose(args) { <br/> &#160; &#160; / code block */ <br/> } |
```

| Triggers before rendering each menu item | Not applicable | **Events:** *beforeItemRender*


```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).BeforeItemRender("beforeItemRender")
.Render() <br/> function beforeItemRender(args) { <br/> &#160; &#160; / code block */ <br/> } |
```

| Triggers while selecting the menu item | **Events:** *click*


```
@Html.EJ().Menu("menu").Click("click") <br/> function click(args) { <br/> &#160; &#160; / code
block / <br/> } | Events: select <br/><br/>
```

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Select("select").Render() <br/> function
select(args) { <br/> &#160; &#160; / code block / <br/> } |
```

| Triggers after closing the menu | **Events:** *close*


```
@Html.EJ().Menu("menu").Close("close") <br/> function close(args) { <br/> &#160; &#160; / code
block / <br/> } | Events: onClose <br/><br/>
```

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).OnClose("onClose").Render() <br/> function
onClose(args) { <br/> &#160; &#160; / code block / <br/> } |
```

| Triggers after opening the menu | **Events:** *open*


```
@Html.EJ().Menu("menu").Open("open") <br/> function open(args) { <br/> &#160; &#160; / code
block / <br/> } | Events: onOpen <br/><br/>
```

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).OnOpen("onOpen").Render() <br/> function
onOpen(args) { <br/> &#160; &#160; / code block / <br/> } |
```

| Triggers once the component rendering is completed | **Events:** *create*


```
@Html.EJ().Menu("menu").Create("create") <br/> function create(args) { <br/> &#160; &#160; /
code block / <br/> } | Events: created <br/><br/>
```

```
@Html.EJS().Menu("menu").Items(ViewBag.menuItems).Created("created").Render() <br/> function
created() { <br/> &#160; &#160; / code block / <br/> } |
```

| Triggers once the component is destroyed | **Events:** *destroy*


```
@Html.EJ().Menu("menu").Destroy("destroy") <br/> function destroy(args) { <br/> &#160; &#160; /
code block */ <br/> } | Not applicable |
```

| Triggers when key down on menu items | **Events:** *keydown*


```
@Html.EJ().Menu("menu").Keydown("keydown") <br/> function keydown(args) { <br/> &#160;
&#160; / code block */ <br/> } | Not applicable |
```

| Triggers when mouse out from menu items | **Events:** *mouseout*

 @Html.EJ().Menu("menu").Mouseout("mouseout")
 function mouseout(args) {

 / code block */
 } | Not applicable |

| Triggers when mouse over the Menu items | **Events:** *mouseover*

 @Html.EJ().Menu("menu").Mouseover("mouseover")
 function mouseover(args) {

 / code block */
 } | Not applicable |

| Triggers when overflow popup menu opens | **Events:** *overflowOpen*

 @Html.EJ().Menu("menu").OverflowOpen("overflowOpen")
 function overflowOpen(args) {

 / code block */
 } | Not applicable |

| Triggers when overflow popup menu closes | **Events:** *overflowClose*

 @Html.EJ().Menu("menu").OverflowClose("overflowClose")
 function overflowClose(args) {

 / code block */
 } | Not applicable |

Message

Getting Started with ASP.NET MVC Message Control

This section briefly explains how to include ASP.NET MVC Message control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

```
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

~/ _LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ _LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```


Add ASP.NET MVC Message control

Now, add the Syncfusion ASP.NET MVC Message control in the `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().Message("msg-default").Content("Please read the comments
carefully").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Message control will be rendered in the default web browser.

 Please read the comments carefully

Severities in Message control

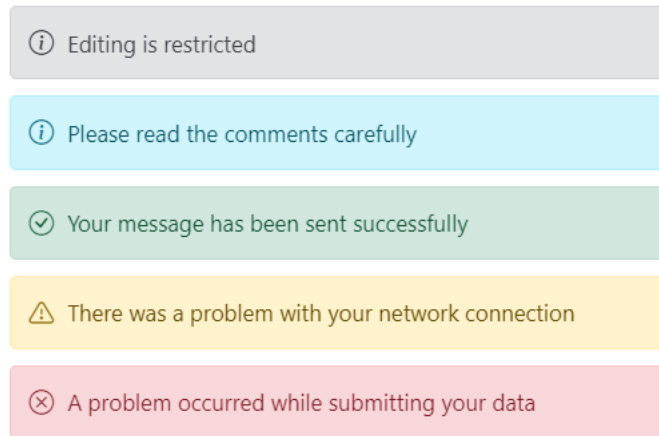
The severity denotes the importance and context of the message to the user. The message contains different severity types. Use the [Severity](#) property to display the messages with different severity levels.

The available severity types are **Normal**, **Success**, **Info**, **Warning** and **Error**. The default severity type for messages is **Normal**.

The following example demonstrates the severity of the messages.

CSHTML

```
<div class="msg-default-section">
  <div class="msg-content-section">
    @Html.EJS().Message("msg-
default").Severity(Severity.Normal).Content("Editing is
restricted").Render()
    @Html.EJS().Message("msg-
info").Severity(Severity.Info).Content("Please read the comments
carefully").Render()
    @Html.EJS().Message("msg-
success").Severity(Severity.Success).Content("Your message has been sent
successfully").Render()
    @Html.EJS().Message("msg-
warning").Severity(Severity.Warning).Content("There was a problem with your
network connection").Render()
    @Html.EJS().Message("msg-error").Severity(Severity.Error).Content("A
problem occurred while submitting your data").Render()
  </div>
</div>
<style>
  .msg-default-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
  }
  .msg-default-section .e-message {
    margin: 10px 0;
  }
</style>
```

Variants in Message control

The Message has predefined appearance variants for different visual representations. The variants of the message can be changed based on the [Variant](#) property.

The available variants are **Text**, **Outlined** and **Filled**. The default variant type for messages is **Text**.

- **Text** - The severity is differentiated using a text color and a light background color.
- **Outlined** - The severity is differentiated using a text color and a border without a background.
- **Filled** - The severity is differentiated using a text color and a dark background color.

The following example demonstrates the default message with different variant types.

CSHTML

```
<div class="msg-variant-section">
  <div class="msg-content-section">
    <h4>Filled</h4>
    @Html.EJS().Message("msg-
default_filled").Severity(Severity.Normal).Variant(Variant.Filled).Content("
Editing is restricted").Render()
    @Html.EJS().Message("msg-
info_filled").Severity(Severity.Info).Variant(Variant.Filled).Content("Pleas
e read the comments carefully").Render()
    @Html.EJS().Message("msg-
success_filled").Severity(Severity.Success).Variant(Variant.Filled).Content(
"Your message has been sent successfully").Render()
    @Html.EJS().Message("msg-
warning_filled").Severity(Severity.Warning).Variant(Variant.Filled).Content(
"There was a problem with your network connection").Render()
    @Html.EJS().Message("msg-
error_filled").Severity(Severity.Error).Variant(Variant.Filled).Content("A
problem occurred while submitting your data").Render()
  </div>
  <div class="msg-content-section">
    <h4>Outlined</h4>
    @Html.EJS().Message("msg-
default_outlined").Severity(Severity.Normal).Variant(Variant.Outlined).Conte
nt("Editing is restricted").Render()
```

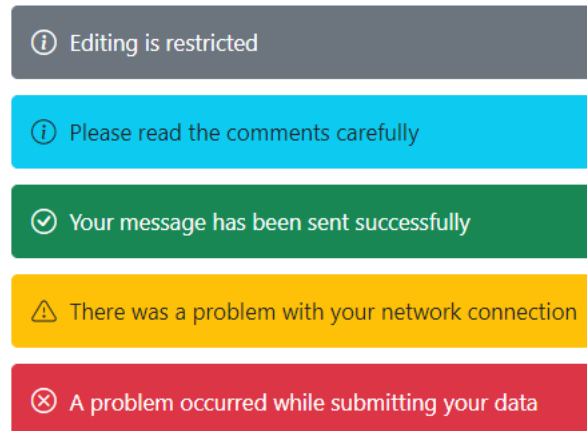
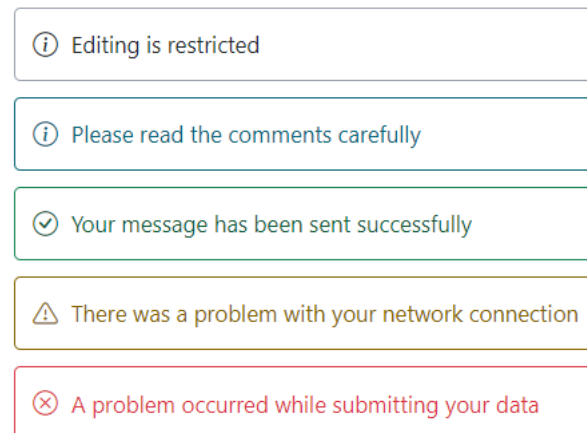
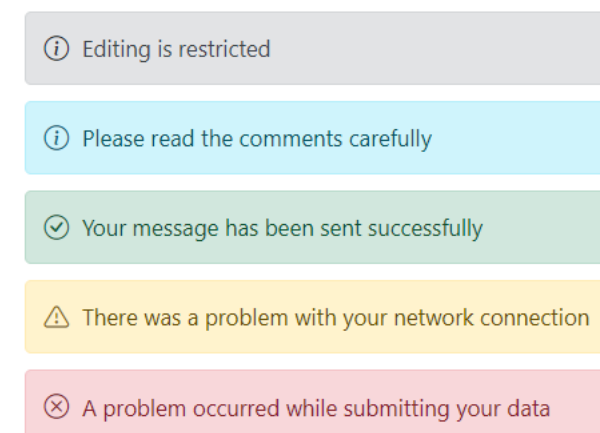
```

        @Html.EJS().Message("msg-
info_outlined").Severity(Severity.Info).Variant(Variant.Outlined).Content("P
lease read the comments carefully").Render()
        @Html.EJS().Message("msg-
success_outlined").Severity(Severity.Success).Variant(Variant.Outlined).Cont
ent("Your message has been sent successfully").Render()
        @Html.EJS().Message("msg-
warning_outlined").Severity(Severity.Warning).Variant(Variant.Outlined).Cont
ent("There was a problem with your network connection").Render()
        @Html.EJS().Message("msg-
error_outlined").Severity(Severity.Error).Variant(Variant.Outlined).Content(
"A problem occurred while submitting your data").Render()
    </div>
    <div class="msg-content-section">
        <h4>Text</h4>
        @Html.EJS().Message("msg-
default").Severity(Severity.Normal).Content("Editing is
restricted").Render()
        @Html.EJS().Message("msg-
info").Severity(Severity.Info).Content("Please read the comments
carefully").Render()
        @Html.EJS().Message("msg-
success").Severity(Severity.Success).Content("Your message has been sent
successfully").Render()
        @Html.EJS().Message("msg-
warning").Severity(Severity.Warning).Content("There was a problem with your
network connection").Render()
        @Html.EJS().Message("msg-error").Severity(Severity.Error).Content("A
problem occurred while submitting your data").Render()
    </div>
</div>
<style>
    .msg-variant-section .content-section {
        margin: 0 auto;
        max-width: 520px;
        padding: 10px;
    }

    .msg-variant-section .e-message {
        margin: 10px 0;
    }

    .msg-variant-section {
        display: flex;
    }
</style>

```

Filled**Outlined****Text**[Icons in Message control](#)

This section explains the message with no icons, how to show or hide the close icon and add the custom severity icon to the message.

No Icon

By default, severity icons can be displayed according to the severity types to make it more understandable to the user by visual information rather than text. To hide the severity icons, set the [ShowIcon](#) property to `false`.

The following example demonstrates the different severity messages without the severity icons.

CSHTML

```
<div class="msg-default-section">
  <div class="msg-content-section">
    @Html.EJS().Message("msg-
default").Severity(Severity.Normal).Content("Editing is
restricted").ShowIcon(false).Render()
    @Html.EJS().Message("msg-
info").Severity(Severity.Info).Content("Please read the comments
carefully").ShowIcon(false).Render()
    @Html.EJS().Message("msg-
success").Severity(Severity.Success).Content("Your message has been sent
successfully").ShowIcon(false).Render()
    @Html.EJS().Message("msg-
warning").Severity(Severity.Warning).Content("There was a problem with your
network connection").ShowIcon(false).Render()
    @Html.EJS().Message("msg-error").Severity(Severity.Error).Content("A
problem occurred while submitting your data").ShowIcon(false).Render()
  </div>
</div>
<style>
.msg-default-section .content-section {
  margin: 0 auto;
  max-width: 450px;
  padding-top: 10px;
}
.msg-default-section .e-message {
  margin: 10px 0;
}
</style>
```

Editing is restricted

Please read the comments carefully

Your message has been sent successfully

There was a problem with your network connection

A problem occurred while submitting your data


```
pAGMABwBuAHMAUgBlAGcAdQBsaGEAcgBNAGUAcwBzAGEAZwBlAF8AaQBjAG8AbgBzAE0AZQBzAHMAYQBNAGUAXwBpAGMABwBuAHMAVgBlAHIAcwBpAG8AbgAgADEALgAwAE0AZQBzAHMAYQBNAGUAXwBpAGMABwBuAHMARgBvAG4AdAAgAGcAZQBzAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMABWF1ZGlvAAAA) format('true');
font-weight: normal;
font-style: normal;
}
.custom.e-message .e-msg-icon::before {
font-family: 'Message_icons';
content: '\e894';
}
</style>
```

🔊 Essential JS 2 is a modern JavaScript UI Controls library built from the ground up to be lightweight, responsive, modular, and touch friendly. It is written in the TypeScript and has no external dependencies. It also includes complete support for Angular, React, Vue, ASP.NET MVC, and ASP.NET Core frameworks.

Close Icon

The message can be rendered with or without the close icon. The close icon is used to hide the message, either by manually clicking the close icon or through keyboard interaction.

By default, the close icon is not rendered in the message. To show the close icon, set the [ShowCloseIcon](#) property to **true**.

In the following example, the messages are rendered with the close icon.

CSHTML

```
<div class="msg-icon-section">
  <div class="msg-content-section">
    @Html.EJS().Button("btn1").Content("Show Default Message").CssClass("e-outline e-primary msg-hidden").Render()

    @Html.EJS().Message("msg_default_icon").Severity(Severity.Normal).Content("Editing is restricted").ShowCloseIcon(true).Closed("defaultClosed").Render()
    @Html.EJS().Button("btn2").Content("Show Info Message").CssClass("e-outline e-primary e-info msg-hidden").Render()

    @Html.EJS().Message("msg_info_icon").Severity(Severity.Info).Content("Please read the comments carefully").ShowCloseIcon(true).Closed("infoClosed").Render()
    @Html.EJS().Button("btn3").Content("Show Success Message").CssClass("e-outline e-primary e-success msg-hidden").Render()

    @Html.EJS().Message("msg_success_icon").Severity(Severity.Success).Content("Your message has been sent successfully").ShowCloseIcon(true).Closed("successClosed").Render()
```

```

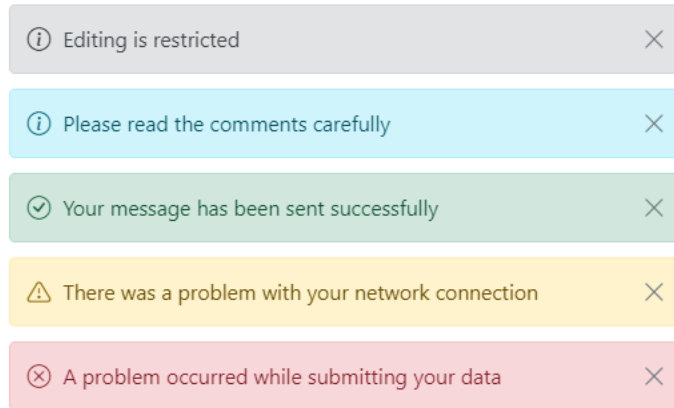
    @Html.EJS().Button("btn4").Content("Show Warning Message").CssClass("e-
outline e-primary e-warning
msg-hidden").Render()

@Html.EJS().Message("msg_warning_icon").Severity(Severity.Warning).Content("
There was a problem with your network
connection").ShowCloseIcon(true).Closed("warningClosed").Render()
    @Html.EJS().Button("btn5").Content("Show Error Message").CssClass("e-
outline e-primary e-danger
msg-hidden").Render()

@Html.EJS().Message("msg_error_icon").Severity(Severity.Error).Content("A
problem has been occurred while submitting
your data").ShowCloseIcon(true).Closed("errorClosed").Render()
</div>
</div>
<script>
    document.getElementById("btn1").onclick = function (e) {
        var msgDefault =
ej.base.getComponent(document.getElementById("msg_default_icon"),
"message");
        var defaultBtn = ej.base.getComponent(document.getElementById("btn1"),
"btn");
        show(msgDefault, defaultBtn);
    }
    document.getElementById("btn2").onclick = function (e) {
        var msgInfo =
ej.base.getComponent(document.getElementById("msg_info_icon"), "message");
        var infoBtn = ej.base.getComponent(document.getElementById("btn2"),
"btn");
        show(msgInfo, infoBtn);
    }
    document.getElementById("btn3").onclick = function (e) {
        var msgSuccess =
ej.base.getComponent(document.getElementById("msg_success_icon"),
"message");
        var successBtn = ej.base.getComponent(document.getElementById("btn3"),
"btn");
        show(msgSuccess, successBtn);
    }
    document.getElementById("btn4").onclick = function (e) {
        var msgWarning =
ej.base.getComponent(document.getElementById("msg_warning_icon"),
"message");
        var warningBtn = ej.base.getComponent(document.getElementById("btn4"),
"btn");
        show(msgWarning, warningBtn);
    }
    document.getElementById("btn5").onclick = function (e) {
        var msgError =
ej.base.getComponent(document.getElementById("msg_error_icon"), "message");
        var errorBtn = ej.base.getComponent(document.getElementById("btn5"),
"btn");
        show(msgError, errorBtn);
    }
</script>
<script>

```

```
function defaultClosed() {
    var defaultBtn = ej.base.getComponent(document.getElementById("btn1"),
    "btn");
    defaultBtn.element.classList.remove("msg-hidden");
}
function infoClosed() {
    var infoBtn = ej.base.getComponent(document.getElementById("btn2"),
    "btn");
    infoBtn.element.classList.remove("msg-hidden");
}
function warningClosed() {
    var warningBtn = ej.base.getComponent(document.getElementById("btn4"),
    "btn");
    warningBtn.element.classList.remove("msg-hidden");
}
function successClosed() {
    var successBtn = ej.base.getComponent(document.getElementById("btn3"),
    "btn");
    successBtn.element.classList.remove("msg-hidden");
}
function errorClosed() {
    var errorBtn = ej.base.getComponent(document.getElementById("btn5"),
    "btn");
    errorBtn.element.classList.remove("msg-hidden");
}
function show(message, btn) {
    message.visible = true;
    btn.element.classList.add("msg-hidden");
}
</script>
<style>
.msg-icon-section .content-section {
    margin: 0 auto;
    max-width: 450px;
    padding-top: 10px;
}
.msg-icon-section .e-message {
    margin: 10px 0;
}
.msg-icon-section .e-btn {
    display: block;
    margin: 10px 0;
}
.msg-icon-section .e-btn.msg-hidden {
    display: none;
}
</style>
```

Customization in Message control

The Message control allows user to customize the content display positions and appearance. This section explains the details about changing the content alignments and border styles for messages.

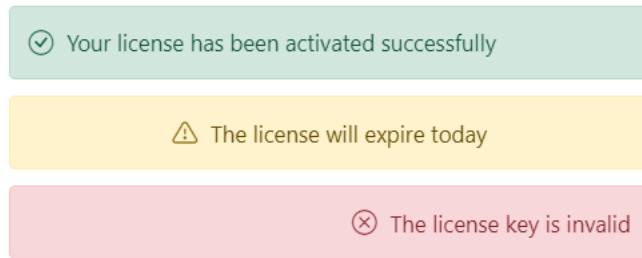
Content Alignment

Normally, the message content is aligned to the **left**. The Message control allows user to align the message content in the **center** or **right** through the built-in classes `e-content-center` and `e-content-right`.

The following example demonstrates the message with different content alignments.

CSHTML

```
<div class="msg-custom-section">
  <div class="msg-content-section">
    <h4>Content Alignment</h4>
    @Html.EJS().Message("msg-content-
left").Severity(Severity.Success).Content("Your license has been activated
successfully").Render()
    @Html.EJS().Message("msg-content-
center").Severity(Severity.Warning).Content("The license will expire
today").CssClass("e-content-center").Render()
    @Html.EJS().Message("msg-content-
right").Severity(Severity.Error).Content("The license key is
invalid").CssClass("e-content-right").Render()
  </div>
</div>
<style>
  .msg-custom-section .content-section {
    margin: 0 auto;
    max-width: 400px;
    padding-top: 10px;
  }
  .msg-custom-section .e-message {
    margin: 10px 0;
  }
</style>
```



Rounded and Square


To customize the Message control's appearance, add the custom class to the message through the [CssClass](#) property. This custom class will be added to the root element. Based on this custom class, the user can override the message styles at the application level.

The following example shows the rounded and squared appearance of the message, which can be achieved by adding the `CssClass` property.

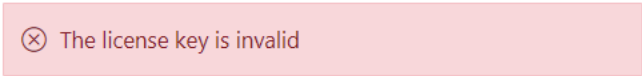
CSHTML

```
<div class="msg-custom-section">
  <div class="msg-content-section">
    <h4>Rounded</h4>
    @Html.EJS().Message("rounded").Severity(Severity.Warning).Content("The
license will expire today").CssClass("rounded").Render()
    <h4>Rounded</h4>
    @Html.EJS().Message("square").Severity(Severity.Error).Content("The
license key is invalid").CssClass("square").Render()
  </div>
</div>
<style>
.msg-custom-section .content-section {
  margin: 0 auto;
  max-width: 400px;
  padding-top: 10px;
}
.msg-custom-section .e-message {
  margin: 10px 0;
}
.msg-custom-section .e-message.rounded {
  border-radius: 5px;
}
.msg-custom-section .e-message.square {
  border-radius: 1px;
}
</style>
```

Rounded



Square



CSS Message

The Essential JS 2 Message has predefined CSS classes that can be defined in the HTML elements, which renders the message without any script reference. This can display a simple message with content and make the code lighter.

The following DOM structure is required to display the simple message with the content.

CSHTML

```
<div class="e-message">
<div class="e-msg-content">..content..</div>
</div>
```

The following DOM structure is required to display the simple message with the content and severity icon.

CSHTML

```
<div class="e-message">
<span class="e-msg-icon"></span>
<div class="e-msg-content">..content..</div>
</div>
```

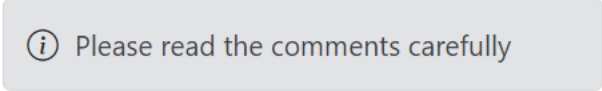
The following is the available list of predefined CSS classes to make the appearance of a message.

| Class | Description |
|------------------|---|
| ----- | ----- |
| e-message | Represents the message wrapper. |
| e-msg-icon | Represents the severity type icon. |
| e-msg-content | Represents the message content. |
| e-msg-close-icon | Represents the close icon. |
| e-info | Represents the information message. |
| e-success | Represents the success message. |
| e-warning | Represents the warning message. |
| e-error | Represents the error message. |
| e-content-center | Aligns the message content to the center. |
| e-content-right | Aligns the message content to the right. |

The following example shows the message which renders without any script reference.

CSHTML

```
<div class="msg-default">
  <div id="msg" class="e-message" role="alert">
    <span class="e-msg-icon"></span>
    <div class="e-msg-content">Please read the comments carefully</div>
  </div>
</div>
<style>
.msg-custom-section .content-section {
  margin: 0 auto;
  max-width: 400px;
  padding-top: 10px;
}
.msg-custom-section .e-message {
  margin: 10px 0;
}
.msg-custom-section .e-message.rounded {
  border-radius: 5px;
}
.msg-custom-section .e-message.square {
  border-radius: 1px;
}
</style>
```



Template in Message control

The message supports templates that allow you to customize the content with a custom structure. The content can be a string, paragraph, or any other HTML element. The template can be rendered through the [Content](#) property or to the [ContentTemplate](#) property.

In the following sample, the Message control content is customized with HTML elements and Syncfusion Button controls, which are directly added to the [ContentTemplate](#) property.

CSHTML

```
<div class="msg-template-section">
<div class="msg-content-section">
@Html.EJS().Button("showBtn").Content("Show pull request").CssClass("e-
outline e-primary msg-hidden").Render()
@Html.EJS().Message("msg_template").Severity(Severity.Success).Created("crea
ted").Closed("closed").ContentTemplate(@<div id="template"><h1>Merged pull
request</h1><p>Pull request #41 merged after a successful build</p><button
id='commitBtn' class='e-control e-btn e-link'>View Commit</button><button
id='btn1' class='e-control e-btn e-link'>Dismiss</button></div>).Render()
</div>
</div>
<script>
document.getElementById("showBtn").onclick = function (e) {
```

```

    var msgTemplate =
ej.base.getComponent(document.getElementById("msg_template"), "message");
    var showBtn = ej.base.getComponent(document.getElementById("showBtn"),
"btn");
    msgTemplate.visible = true;
    showBtn.element.classList.add("msg-hidden");
}
</script>
<script>
function created() {
    document.getElementById("btn1").onclick = function (e) {
        var msgTemplate =
ej.base.getComponent(document.getElementById("msg_template"), "message");
        msgTemplate.visible = false;
    }
}
function closed() {
    var showBtn = ej.base.getComponent(document.getElementById("showBtn"),
"btn");
    showBtn.element.classList.remove("msg-hidden");
}
</script>
<style>
.msg-template-section .content-section {
margin: 0 auto;
max-width: 450px;
padding-top: 20px;
}
.msg-template-section .e-btn.msg-hidden {
display: none;
}
.msg-template-section .e-message h1 {
margin: 0;
font-size: 16px;
font-weight: 600;
line-height: 1.25;
}
.msg-template-section .e-message .e-msg-icon {
padding: 0 4px;
margin-top: 3px;
}
.msg-template-section .e-message p {
margin: 8px 0 4px;
}
.msg-template-section .e-message .e-btn {
padding: 0;
}
</style>

```

✓ Merged pull request

Pull request #41 merged after a successful build

[View commit](#) [Dismiss](#)

Accessibility in ASP.NET MVC Message control

The Message control followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Message control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support | |

| [Section 508](#) Support | |

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| [Accessibility Checker](#) Validation | |

| [Axe-core](#) Accessibility Validation | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

WAI-ARIA attributes

The Message control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Message control:

| Attributes | Purpose |

| --- | --- |

| **role=alert** | Used to convey a significant and contextual message to the user. |

| **aria-label** | Provides an accessible name for the close icon. |

Keyboard interaction

The Message control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message control.

| Press | To do this |

| --- | --- |

| Tab / Shift + Tab | To focus the close icon in the message. |

| Enter / Space | Closes the focused close icon's message. |

Ensuring accessibility

The Message control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Message control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Message control with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC controls](#)

MultiSelect

Getting Started with ASP.NET MVC MultiSelect Control

This section briefly explains about how to include [ASP.NET MVC MultiSelect](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
```



```
</body>
```

Add ASP.NET MVC MultiSelect control

Now, add the Syncfusion ASP.NET MVC MultiSelect control in `~/Views/Home/Index.cshtml` page.

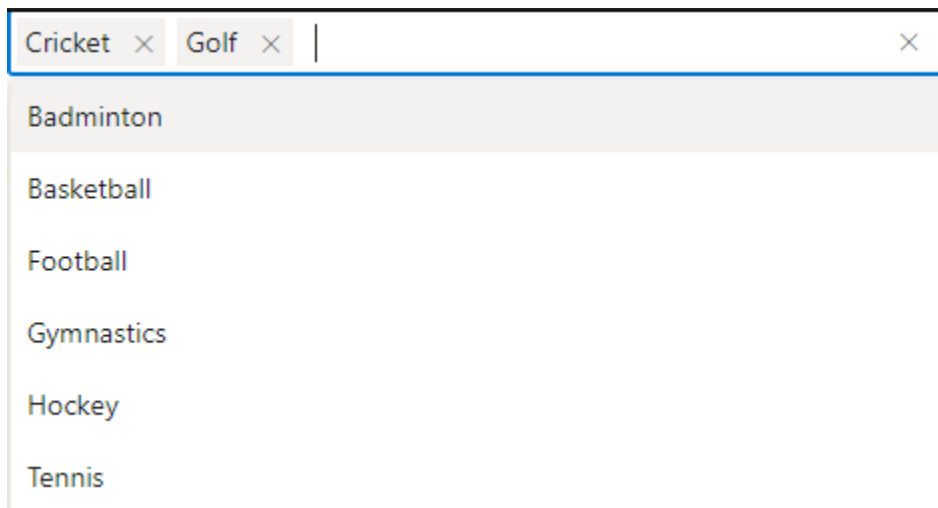
CSHTML

```
@model List<string>
@Html.EJS().MultiSelect("default").DataSource((IEnumerable<object>)Model).Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC MultiSelect control will be rendered in the default web browser.



Configure the popup list

By default, the width of the popup list automatically adjusts according to the MultiSelect input element's width, and the height auto adjust's according to the height of the popup list items.

The height and width of the popup list can also be customized using the [popupHeight](#) and [popupWidth](#) properties respectively.

CSHTML

```
@model List<string>
@Html.EJS().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>)Model).PopupHeight("220px").PopupWidth("300px").Render()
```

HOMECONTROLLER.CS

```
public ActionResult Index()
{
    List<string> data = new List<string>() { "Badminton", "Basketball",
    "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
    return View(data);
}
```

Note: [View Sample in GitHub.](#)

See also

- [How to bind the data](#)

Data Binding

The MultiSelect loads the data either from local data sources or remote data services using the [dataSource](#) property. It supports the data type of array or [DataManager](#).

The MultiSelect also supports different kinds of data services such as OData, OData V4, and Web API, and data formats such as XML, JSON, and JSONP with the help of [DataManager](#) adaptors.

| Fields | Type | Description |
|---------|------------------|--|
| text | string | Specifies the display text of each list item. |
| value | number or string | Specifies the hidden data value mapped to each list item that should contain a unique value. |
| groupBy | string | Specifies the category under which the list item has to be grouped. |
| iconCss | string | Specifies the icon class of each list item. |

Note: When binding complex data to the MultiSelect, fields should be mapped correctly. Otherwise, the selected item remains undefined.

Binding local data

Local data can be represented in two ways as described below.

1. Array of string

The MultiSelect has support to load array of primitive data such as strings and numbers. Here, both value and text field act the same.

CSHTML

```
@Html.EJS().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>) ViewBag.data).Render()
```

ARRAYOFSTRINGS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```

using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult arrayofstrings()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball",
            "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}

```

2. Array of object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, **Id** column and **Game** column from complex data have been mapped to the **value** field and **text** field, respectively.

CSHTML

```

@Html.EJ2().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>)ViewBag.data).Mode(Syncfusion.EJ2.Dr
opDowns.VisualMode.Box).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "Game", Value =
"Id" }).Render()

```

GAMELIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class GameList
    {
        public string Id { get; set; }
        public string Game { get; set; }
        public List<GameList> GameLists()
        {
            List<GameList> game = new List<GameList>();
            game.Add(new GameList { Id = "Game1", Game = "American Football"
        });
            game.Add(new GameList { Id = "Game2", Game = "Badminton" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game4", Game = "Cricket" });
            game.Add(new GameList { Id = "Game5", Game = "Football" });
            game.Add(new GameList { Id = "Game6", Game = "Golf" });
            game.Add(new GameList { Id = "Game7", Game = "Hockey" });
            game.Add(new GameList { Id = "Game8", Game = "Rugby" });
        }
    }
}

```

```

        game.Add(new GameList { Id = "Game9", Game = "Snooker" });
        game.Add(new GameList { Id = "Game10", Game = "Tennis" });
        return game;
    }
}

```

3. Array of complex object

The MultiSelect can generate its list items through an array of complex data. For this, the appropriate columns should be mapped to the [fields](#) property.

In the following example, `Code.Id` column and `Country.CountryId` column from complex data have been mapped to the `value` field and `text` field, respectively.

CSHTML

```

@Html.EJS().MultiSelect("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<Object>)ViewBag.data).Fi
elds(new Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text =
"Country.CountryId", Value = "Code.Id" }).Render()

```

COMPLEX.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Code
    {
        public string Id { get; set; }
    }
    public class Country
    {
        public string CountryId { get; set; }
    }
    public class Complex
    {
        public Country Country { get; set; }
        public Code Code { get; set; }
        public List<Complex> GetData()
        {
            List<Complex> data = new List<Complex>();
            data.Add(new Complex() { Country = new Country() { CountryId =
"Australia" }, Code = new Code() { Id = "AU" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Bermuda" }, Code = new Code() { Id = "BM" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Canada" }, Code = new Code() { Id = "CA" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Cameroon" }, Code = new Code() { Id = "CM" } });
            data.Add(new Complex() { Country = new Country() { CountryId =
"Denmark" }, Code = new Code() { Id = "DK" } });
        }
    }
}

```

```

        data.Add(new Complex() { Country = new Country() { CountryId =
"France" }, Code = new Code() { Id = "FR" } });
        return data;
    }
}

```

Binding remote data

The MultiSelect supports retrieval of data from remote data services with the help of **DataManager** control. The **Query** property is used to fetch data from the database and bind it to the MultiSelect.

The following sample displays the first 6 contacts from “Customers” table of the **Northwind** Data Service.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Text = "ContactName",
    Value = "CustomerID"
}).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()

```

REMOTEDATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult remotedata()
        {
            return View();
        }
    }
}

```

Bind to URL Adaptor

The MultiSelect supports retrieval of data from URL adaptor.

CSHTML

```

@Html.EJS().MultiSelect("games").Placeholder("Select a
Country").PopupHeight("200px").DataSource(dataManger =>

```

```
dataManger.Url("/MultiSelect/UrlDatasource").Adaptor("UrlAdaptor").CrossDomain(true)).Fields(new Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Value = "ShipCountry"
}).Render()
```

URLDATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiselectController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
        public ActionResult UrlDatasource([FromBody]Data dm)
        {
            var val = OrdersDetails.GetAllRecords();
            var Data = val.ToList();
            var count = val.Count();
            if (dm.where != null)
            {
                Data = (from cust in Data
                        where
cust.ShipCountry.ToLower().StartsWith(dm.@where[0].value.ToString())
                        select cust).ToList();
            }
            if (dm.take != 0)
                Data = Data.Take(dm.take).ToList();
            return Json(Data);
        }
        public class Data
        {
            public int take { get; set; }
            public List<Wheres> where { get; set; }
        }
        public class Wheres
        {
            public string field { get; set; }
            public bool ignoreAccent { get; set; }
            public bool ignoreCase { get; set; }
            public bool isComplex { get; set; }
            public string value { get; set; }
            public string Operator { get; set; }
        }
        public class OrdersDetails
        {

```

```

        public static List<OrdersDetails> order = new
List<OrdersDetails>();
        public OrdersDetails()
        {
        }
        public OrdersDetails(int OrderID, string CustomerId, int
EmployeeId, double Freight, bool Verified, DateTime OrderDate, string
ShipCity, string ShipName, string ShipCountry, DateTime ShippedDate, string
ShipAddress)
        {
            this.OrderID = OrderID;
            this.CustomerID = CustomerId;
            this.EmployeeID = EmployeeId;
            this.Freight = Freight;
            this.ShipCity = ShipCity;
            this.Verified = Verified;
            this.OrderDate = OrderDate;
            this.ShipName = ShipName;
            this.ShipCountry = ShipCountry;
            this.ShippedDate = ShippedDate;
            this.ShipAddress = ShipAddress;
        }
        public static List<OrdersDetails> GetAllRecords()
        {
            if (order.Count() == 0)
            {
                int code = 10000;
                for (int i = 1; i < 10; i++)
                {
                    order.Add(new OrdersDetails(code + 1, "ALFKI", i +
0, 2.3 * i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro",
"Denmark", new DateTime(1996, 7, 16), "Kirchgasse 6"));
                    order.Add(new OrdersDetails(code + 2, "ANATR", i +
2, 3.3 * i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha",
"Brazil", new DateTime(1996, 9, 11), "Avda. Azteca 123"));
                    order.Add(new OrdersDetails(code + 3, "ANTON", i +
1, 4.3 * i, true, new DateTime(1957, 11, 30), "Cholchester",
"Frankenversand", "Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave.
Bolívar #65-98 Llano Largo"));
                    order.Add(new OrdersDetails(code + 4, "BLONP", i +
3, 5.3 * i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel",
"Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
                    order.Add(new OrdersDetails(code + 5, "BOLID", i +
4, 6.3 * i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes",
"Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
                    code += 5;
                }
            }
            return order;
        }
        public int? OrderID { get; set; }
        public string CustomerID { get; set; }
        public int? EmployeeID { get; set; }
        public double? Freight { get; set; }
        public string ShipCity { get; set; }
        public bool Verified { get; set; }
        public DateTime OrderDate { get; set; }

```

```

        public string ShipName { get; set; }
        public string ShipCountry { get; set; }
        public DateTime ShippedDate { get; set; }
        public string ShipAddress { get; set; }
    }
}
}

```

Web API Adaptor

Use the **WebApiAdaptor** to bind MultiSelect with Web API created using OData.

CSHTML

```

<div id='web-data' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>
        @Html.EJS().MultiSelect("games").Placeholder("Select
customer").DataSource(dataManger =>

dataManger.Url("/api/MultiSelect/").Adaptor("WebApiAdaptor")).Render()
    </div>
</div>

```

WEBAPI.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    [Route("api/[controller]")]
    public class MultiselectController : Controller
    {
        List<string> game = new List<string>();
        [HttpGet]
        public List<string> Get()
        {
            game.Add("Badminton");
            game.Add("Basketball");
            game.Add("Cricket");
            game.Add("Golf");
            game.Add("Gymnastics");
            game.Add("Tennis");
            game.Add("Hockey");
            return game;
        }
    }
}

```


Binding with OData services

OData is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the **DataManager**.

The following example for remote data binding using OData service.

CSHTML

```
<div id='o-data' class='col-lg-6' style='padding-top:15px'>
  <div class='content'>
    @Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Order
s/").Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
    {
        Value = "CustomerID"
    }).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
  </div>
</div>
```

ODATA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiselectController : Controller
    {
        public ActionResult odata()
        {
            return View();
        }
    }
}
```

Offline mode

To avoid post back for every action, set the MultiSelect to load all data on initialization and make the actions process in client-side. To enable this behavior, use the **Offline** property of **DataManager**.

The following example for remote data binding and enabled offline mode.

CSHTML

```
<div id='o-data' class='col-lg-6' style='padding-top:15px'>
  <div class='content'>
    @Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("http://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Orders
```

```

/").Offline(true).Adaptor("ODataAdaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Value = "CustomerID"
}).Query("new
ej.data.Query().select(['CustomerID']).take(7)").Render()
</div>
</div>

```

OFFLINE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiselectController : Controller
    {
        public ActionResult offline()
        {
            return View();
        }
    }
}

```

See Also

- [How to load data using template](#)
- [How to group the data using header](#)
- [How to filter the bound data](#)

Templates in MultiSelect Control

The MultiSelect has been provided with several options to customize each list item, group title, selected value, header, and footer elements. It uses the Essential JS 2 **Template engine** to compile and render the elements properly.

Item template

The content of each list item within the MultiSelect can be customized with the help of [itemTemplate](#) property.

In the following sample, each list item is split into two columns to display relevant data's.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("O
DataV4Adaptor").CrossDomain(true)).ItemTemplate("<span><span
class='name'>${FirstName}</span><span class
='city'>${City}</span></span>").Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings

```

```

        {
            Value = "FirstName"
        }).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
    .city {
        right: 15px;
        position: absolute;
    }
</style>

```

ITEMTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult Itemtemplate()
        {
            return View();
        }
    }
}

```

Value template

The currently selected value that is displayed by default on the MultiSelect input element can be customized using the [valueTemplate](#) property.

In the following sample, the selected value is displayed as a combined text of both **FirstName** and **City** in the MultiSelect input, which is separated by a hyphen.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("<span><span
class='name'>${FirstName}</span><span class
='city'>${City}</span></span>").ValueTemplate("<span>${FirstName} -
${City}</span>").Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Value = "FirstName",
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>

```

```
.city {
    right: 15px;
    position: absolute;
}
</style>
```

VALUETEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult valuetemplate()
        {
            return View();
        }
    }
}
```

Group template

The group header title under which appropriate sub-items are categorized can also be customize with the help of [groupTemplate](#) property. This template is common for both inline and floating group header template.

In the following sample, employees are grouped according to their city.

CSHTML

```
@Html.EJS().MultiSelect("customers").Placeholder("Select a customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor("ODataV4Adaptor").CrossDomain(true)).GroupTemplate("<strong>${City}</strong>").Fields(new Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Value = "FirstName",
    GroupBy = "City"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City', 'EmployeeID']).take(6).where(new ej.data.Predicate('City', 'equal', 'london').or('City', 'equal', 'seattle'))").Render()
```

GROUPTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

```
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult grouptemplate()
        {
            return View();
        }
    }
}
```

Header template

The header element is shown statically at the top of the popup list items within the MultiSelect, and any custom element can be placed as a header element using the [headerTemplate](#) property.

In the following sample, the list items and its headers are designed and displayed as two columns similar to multiple columns of the grid.

CSHTML

```
@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).ItemTemplate("<span class='item'
><span class='name'>${FirstName}</span><span
class='city'>${City}</span></span>").HeaderTemplate("<span
class='head'><span class='name'>Name</span><span
class='city'>City</span></span>").Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Value = "FirstName",
}).Query("new
ej.data.Query().from('Employees').select(['FirstName', 'City',
'EmployeeID']).take(6)").Render()
<style>
    .head, .item {
        display: table;
        width: 100%;
        margin: auto;
    }
    .head {
        height: 40px;
        font-size: 15px;
        font-weight: 600;
    }
    .name, .city {
        display: table-cell;
        vertical-align: middle;
        width: 50%;
    }
    .head .name {
        text-indent: 16px;
    }
    .head .city {
```

```

        text-indent: 10px;
    }
</style>

```

HEADERTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult headertemplate()
        {
            return View();
        }
    }
}

```

Footer template

The MultiSelect has options to show a footer element at the bottom of the list items in the popup list. Here, you can place any custom element as a footer element using the [footerTemplate](#) property.

In the following sample, footer element displays the total number of list items present in the MultiSelect.

CSHTML

```

@Html.EJS().MultiSelect("games").DataSource((IEnumerable<object>)ViewBag.data).Placeholder("Select a game").PopupHeight("270px").FooterTemplate("<span class='foot'> Total list items: " + 5 + "</span>").Render()
<style>
    .foot {
        text-indent: 1.2em;
        display: block;
        font-size: 15px;
        line-height: 40px;
        border-top: 1px solid #e0e0e0;
    }
</style>

```

FOOTERTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{

```

```

public class MultiSelectController : Controller
{
    public ActionResult footertemplate()
    {
        ViewBag.data = new string[] { "Badminton", "Basketball",
"Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
        return View();
    }
}

```

No records template

The MultiSelect is provided with support to custom design the popup list content when no data is found and no matches found on search with the help of [noRecordsTemplate](#) property.

In the following sample, popup list content displays the notification of no data available.

CSHTML

```

@Html.EJS().MultiSelect("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).No
RecordsTemplate("<span class='norecord'> NO DATA AVAILABLE</span>").Render()

```

NORECORDSTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult norecords()
        {
            ViewBag.data = new string[] { };
            return View();
        }
    }
}

```

Action failure template

There is also an option to custom design the popup list content when the data fetch request fails at the remote server. This can be achieved using the [actionFailureTemplate](#) property.

In the following sample, when the data fetch request fails, the MultiSelect displays the notification.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svcs/").Ad

```

```
aptor("ODataV4Adaptor").CrossDomain(true)).ActionFailureTemplate("<span
class='action-failure'> Data fetch get fails</span>").Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
{
    Value = "FirstName"
}).Query("new
ej.data.Query().from('Employees').select(['FirstName']).take(6)").Render()
```

ACTIONFAILURETEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult actionfailure()
        {
            return View();
        }
    }
}
```

See Also

- [How to bind the data](#)
- [How to group the data using header](#)
- [How to customize the options in MultiSelect](#)

Grouping

The MultiSelect supports wrapping nested elements into a group based on different categories. The category of each list item can be mapped through the [groupBy](#) field in the data table. The group header is displayed both as inline and fixed headers. The fixed group header content is updated dynamically on scrolling the popup list with its category value.

In the following sample, vegetables are grouped according on its category using `groupBy` field.

CSHTML

```
@Html.EJS().MultiSelect("Vegetable").Placeholder("Select a
Vegetable").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.dat
a).Fields(new Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Value =
"Vegetable", GroupBy = "Category" }).Render()
```

GROUPING.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
```



```

using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult grouping()
        {
            ViewBag.data = new Vegetables().VegetablesList();
            return View();
        }
    }
}

```

Customization

The grouping header is also provided with customization option. This allows custom designing using the `groupTemplate` property for both inline and fixed headers as referred here: [Group Template support to MultiSelect](#).

Grouping with CheckBox

Previously, there is no checkbox for group headers. Now, this feature allow to render checkbox in group header to select the group items in single selection. You can enable this feature by setting `enableGroupCheckBox` property value as **true** and `mode` property as **CheckBox**.

Inject the `CheckBoxSelection` module in the MultiSelect to use the checkbox.

CSHTML

```

@Html.EJS().MultiSelect("Vegetable").Placeholder("Select
Vegetables").PopupHeight("200px").Mode(Syncfusion.EJ2.DropDowns.VisualMode.C
heckBox).ShowSelectAll(true).EnableGroupCheckBox(true).AllowFiltering(true).
FilterBarPlaceholder("Search
Vegetables").DataSource((IEnumerable<object>)ViewBag.data).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Value = "Vegetable",
GroupBy = "Category" }).Render()

```

GROUPING.CS

```

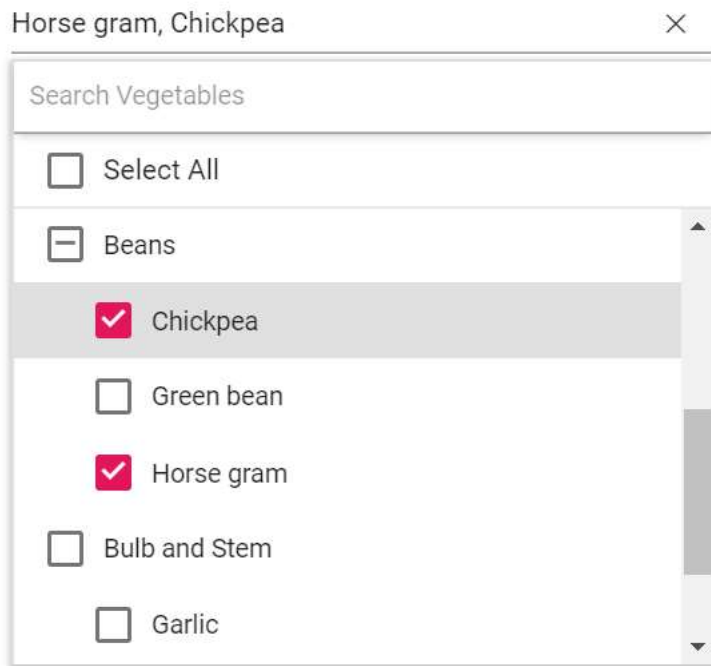
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult grouping()
        {
            ViewBag.data = new Vegetables().VegetablesList();
            return View();
        }
    }
}

```

```
}

```

Output be like the below:



Filtering

The MultiSelect has built-in support to filter data items when [allowFiltering](#) is enabled. The filter operation starts as soon as you start typing characters in the MultiSelect input.

To display filtered items in the popup, filter the required data and return it to the MultiSelect via `updateData` method by using the [filtering](#) event.

The following sample illustrates how to query the data source and pass the data to the MultiSelect through the `updateData` method in [filtering](#) event.

CSHTML

```
@Html.EJS().MultiSelect("country").Placeholder("Select a
Country").PopupHeight(
    "230px").DataSource((IEnumerable<object>)ViewBag.data).AllowFiltering(true).
Filtering("onfiltering").Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Value = "Name"
}).Render()
<script>
    function onfiltering(e) {
        var query = new ej.data.Query();
        query = (e.text !== '') ? query.where('Name', 'startswith',
e.text, true) : query;
e.updateData(@Html.Raw(JsonConvert.SerializeObject(ViewBag.data)), query)
    }
</script>
```

COUNTRIES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Countries
    {
        public string Name { get; set; }
        public string Code { get; set; }
        public List<Countries> CountriesList()
        {
            List<Countries> country = new List<Countries>();
            country.Add(new Countries { Name = "Australia", Code = "AU" });
            country.Add(new Countries { Name = "Bermuda", Code = "BM" });
            country.Add(new Countries { Name = "Canada", Code = "CA" });
            country.Add(new Countries { Name = "Cameroon", Code = "CM" });
            country.Add(new Countries { Name = "Denmark", Code = "DK" });
            country.Add(new Countries { Name = "France", Code = "FR" });
            country.Add(new Countries { Name = "Finland", Code = "FI" });
            country.Add(new Countries { Name = "Germany", Code = "DE" });
            country.Add(new Countries { Name = "Greenland", Code = "GL" });
            country.Add(new Countries { Name = "Hong Kong", Code = "HK" });
            country.Add(new Countries { Name = "India", Code = "IN" });
            country.Add(new Countries { Name = "Italy", Code = "IT" });
            country.Add(new Countries { Name = "Japan", Code = "JP" });
            country.Add(new Countries { Name = "Mexico", Code = "MX" });
            country.Add(new Countries { Name = "Norway", Code = "NO" });
            country.Add(new Countries { Name = "Poland", Code = "PL" });
            country.Add(new Countries { Name = "Switzerland", Code = "CH"
        });
            country.Add(new Countries { Name = "United Kingdom", Code = "GB"
        });
            country.Add(new Countries { Name = "United States", Code = "US"
        });
            return country;
        }
    }
}

```

Limit the minimum filter character

When filtering the list items, you can set the limit for character count to raise remote request and fetch filtered data on the MultiSelect. This can be done by manual validation within the filter event handler.

In the following example, the remote request does not fetch the search data until the search key contains three characters.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").Filtering("onfiltering").AllowFiltering(true).PopupHeight("200px"
).DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(

```

```

        "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "ContactName",
Value = "CustomerID" }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
    function onfiltering(e) {
        var CBObj = document.getElementById("customers").ej2_instances[0];
        // load overall data when search key empty.
        if (e.text == '' && e.text.length < 3) {
            e.updateData(CBObj.dataSource);
        }
        let query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
        query = (e.text != '' && e.text.length >= 3) ?
query.where('ContactName', 'startswith', e.text, true) : query;
        e.updateData(CBObj.dataSource, query);
    }
</script>

```

FILTERLIMIT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult filterlimit()
        {
            return View();
        }
    }
}

```

Change the filter type

While filtering, you can change the filter type to `contains`, `startsWith`, or `endsWith` for string type within the filter event handler.

In the following examples, data filtering is done with `endsWith` type.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").AllowFiltering(true).Filtering("onfiltering").PopupHeight("200px")
.DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
    "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "ContactName",
Value = "CustomerID" }).Query("new

```

```

ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>
    function onfiltering(e) {
        var CBObj = document.getElementById("customers").ej2_instances[0];
        // load overall data when search key empty.
        if (e.text == '')
            e.updateData(CBObj.dataSource);
        else {
            var query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
            query = (e.text != '') ? query.where('ContactName', 'endswith',
e.text, true) : query;
            e.updateData(CBObj.dataSource, query);
        }
    }
</script>

```

FILTERTYPE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult filtertype()
        {
            return View();
        }
    }
}

```

Case sensitive filtering

Data items can be filtered either with or without case sensitivity using the DataManager. This can be done by passing the fourth optional parameter of the `where` clause.

The following example shows how to perform case-sensitive filter.

CSHTML

```

@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").Filtering("onfiltering").AllowFiltering(true).PopupHeight("200px")
).DataSource(obj =>
obj.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Adaptor(
    "ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "ContactName",
Value = "CustomerID" }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6)").Render()
<script>

```

```
function onfiltering(e) {
    var CBObj = document.getElementById("customers").ej2_instances[0];
    // load overall data when search key empty.
    if (e.text == '')
        e.updateData(CBObj.dataSource);
    else {
        var query = new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(6);
        query = (e.text != '') ? query.where('ContactName',
'startswith', e.text, false) : query;
        e.updateData(CBObj.dataSource, query);
    }
}
</script>
```

CASESENSITIVE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

Diacritics Filtering

MultiSelect supports diacritics filtering which will ignore the [diacritics](#) and makes it easier to filter the results in international characters lists when the [ignoreAccent](#) is enabled.

In the following sample, data with diacritics are bound as dataSource for MultiSelect.

CSHTML

```
@Html.EJS().MultiSelect("list").DataSource((string[])ViewBag.data).Placeholder("e.g: aero").IgnoreAccent(true).AllowFiltering(true).Render()
```

DIACRITICS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
```

```
{
    public class MultiSelectController : Controller
    {
        public ActionResult diacritics()
        {
            ViewBag.data = new string[] { "Aeróbics", "Aeróbics en Agua",
            "Aerografía", "Aeromodelaje", "Águilas", "Ajedrez", "Ala Delta", "Álbumes de
            Música", "Alusivos", "Análisis de Escritura a Mano" };
            return View();
        }
    }
}
```

See Also

- [How to bind the data](#)
- [How to group the data using header](#)
- [How to add custom value to the MultiSelect](#)

CustomValue

The MultiSelect allows user to add a new non-present option to the control value when [allowCustomValue](#) is enabled. while selecting the new custom value `customValueSelection` event will be triggered.

The following sample demonstrates configuration of custom value support with the MultiSelect control.

CSHTML

```
<div id='local-data'>
    <div class='control-wrapper'>
        @Html.EJS().MultiSelect("default").Placeholder("Select
        games").AllowCustomValue(true).DataSource((IEnumerable<object>)ViewBag.data)
        .Fields(new Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text =
        "Game", Value = "Id" }).Render()
    </div>
</div>
```

GAMELIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class GameList
    {
        public string Id { get; set; }
        public string Game { get; set; }
        public List<GameList> GameLists()
        {
            List<GameList> game = new List<GameList>();
            game.Add(new GameList { Id = "Game1", Game = "American Football"
        });
    }
}
```

```

        game.Add(new GameList { Id = "Game2", Game = "Badminton" });
        game.Add(new GameList { Id = "Game3", Game = "Basketball" });
        game.Add(new GameList { Id = "Game3", Game = "Basketball" });
        game.Add(new GameList { Id = "Game4", Game = "Cricket" });
        game.Add(new GameList { Id = "Game5", Game = "Football" });
        game.Add(new GameList { Id = "Game6", Game = "Golf" });
        game.Add(new GameList { Id = "Game7", Game = "Hockey" });
        game.Add(new GameList { Id = "Game8", Game = "Rugby" });
        game.Add(new GameList { Id = "Game9", Game = "Snooker" });
        game.Add(new GameList { Id = "Game10", Game = "Tennis" });
        return game;
    }
}

```

CheckBox in MultiSelect Control

The MultiSelect has built-in support to select multiple values through checkbox, when [mode](#) property set as **CheckBox**.

To use checkbox, inject the **CheckBoxSelection** module in the MultiSelect.

CSHTML

```

@Html.EJS().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>)ViewBag.data).Mode(Syncfusion.EJ2.Dr
opDowns.VisualMode.CheckBox).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "Game", Value =
"Id" }).Render()

```

GAMELIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class GameList
    {
        public string Id { get; set; }
        public string Game { get; set; }
        public List<GameList> GameLists()
        {
            List<GameList> game = new List<GameList>();
            game.Add(new GameList { Id = "Game1", Game = "American Football"
});
            game.Add(new GameList { Id = "Game2", Game = "Badminton" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game4", Game = "Cricket" });
            game.Add(new GameList { Id = "Game5", Game = "Football" });
            game.Add(new GameList { Id = "Game6", Game = "Golf" });
            game.Add(new GameList { Id = "Game7", Game = "Hockey" });
            game.Add(new GameList { Id = "Game8", Game = "Rugby" });
            game.Add(new GameList { Id = "Game9", Game = "Snooker" });
            game.Add(new GameList { Id = "Game10", Game = "Tennis" });

```



```

        return game;
    }
}

```

Select All

The MultiSelect control has in-built support to select the all list items using **Select All** options in the header.

When the [showSelectAll](#) property is set to true, by default Select All text will show. You can customize the name attribute of the Select All option by using [selectAllText](#).

For the unSelect All option, by default unSelect All text will show. You can customize the name attribute of the unSelect All option by using [unselectAllText](#).

CSHTML

```

@Html.EJS().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>)ViewBag.data).Mode(Syncfusion.EJ2.Dr
opDowns.VisualMode.CheckBox).ShowSelectAll(true).SelectAllText("Select
All").UnSelectAllText("UnSelect All").Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "Game", Value =
"Id" }).Render()

```

GAMELIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class GameList
    {
        public string Id { get; set; }
        public string Game { get; set; }
        public List<GameList> GameLists()
        {
            List<GameList> game = new List<GameList>();
            game.Add(new GameList { Id = "Game1", Game = "American Football"
});
            game.Add(new GameList { Id = "Game2", Game = "Badminton" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game4", Game = "Cricket" });
            game.Add(new GameList { Id = "Game5", Game = "Football" });
            game.Add(new GameList { Id = "Game6", Game = "Golf" });
            game.Add(new GameList { Id = "Game7", Game = "Hockey" });
            game.Add(new GameList { Id = "Game8", Game = "Rugby" });
            game.Add(new GameList { Id = "Game9", Game = "Snooker" });
            game.Add(new GameList { Id = "Game10", Game = "Tennis" });
            return game;
        }
    }
}

```

Selection Limit

Defines the limit of the selected items using [maximumSelectionLength](#).

CSHTML

```
@Html.EJS().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>)ViewBag.data).Mode(Syncfusion.EJ2.Dr
opDowns.VisualMode.CheckBox).MaximumSelectionLength(3).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "Game", Value =
"Id" }).Render()
```

GAMELIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class GameList
    {
        public string Id { get; set; }
        public string Game { get; set; }
        public List<GameList> GameLists()
        {
            List<GameList> game = new List<GameList>();
            game.Add(new GameList { Id = "Game1", Game = "American Football"
});
            game.Add(new GameList { Id = "Game2", Game = "Badminton" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game4", Game = "Cricket" });
            game.Add(new GameList { Id = "Game5", Game = "Football" });
            game.Add(new GameList { Id = "Game6", Game = "Golf" });
            game.Add(new GameList { Id = "Game7", Game = "Hockey" });
            game.Add(new GameList { Id = "Game8", Game = "Rugby" });
            game.Add(new GameList { Id = "Game9", Game = "Snooker" });
            game.Add(new GameList { Id = "Game10", Game = "Tennis" });
            return game;
        }
    }
}
```

Selection Reordering

Using [enableSelectionOrder](#) to Reorder the selected items in popup visibility state.

CSHTML

```
@Html.EJS().MultiSelect("default").Placeholder("Select
games").DataSource((IEnumerable<object>)ViewBag.data).Mode(Syncfusion.EJ2.Dr
opDowns.VisualMode.CheckBox).EnableSelectionOrder(false).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "Game", Value =
"Id" }).Render()
```

GAMELIST.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class GameList
    {
        public string Id { get; set; }
        public string Game { get; set; }
        public List<GameList> GameLists()
        {
            List<GameList> game = new List<GameList>();
            game.Add(new GameList { Id = "Game1", Game = "American Football"
});
            game.Add(new GameList { Id = "Game2", Game = "Badminton" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game3", Game = "Basketball" });
            game.Add(new GameList { Id = "Game4", Game = "Cricket" });
            game.Add(new GameList { Id = "Game5", Game = "Football" });
            game.Add(new GameList { Id = "Game6", Game = "Golf" });
            game.Add(new GameList { Id = "Game7", Game = "Hockey" });
            game.Add(new GameList { Id = "Game8", Game = "Rugby" });
            game.Add(new GameList { Id = "Game9", Game = "Snooker" });
            game.Add(new GameList { Id = "Game10", Game = "Tennis" });
            return game;
        }
    }
}

```

See Also

- [How to bind the data](#)
- [How to filter the bound data](#)
- [How to add custom value to the MultiSelect](#)
- [How to render checkbox in grouping to the MultiSelect.](#)

Chip Customization

The MultiSelect allows the user to customize the selected chip element through the [tagging](#) event. In that event, you can set the custom classes to chip element via that event argument of `setClass` method.

The following sample demonstrates chip-customization with the MultiSelect control.

CSHTML

```

<div id='local-data'>
    <div class='control-wrapper'>
        @Html.EJS().MultiSelect("chip-
customization").Tagging("onTagging").Placeholder("Select a
color").DataSource((IEnumerable<object>)ViewBag.data).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Text = "Color", Value =
"Code" }).Render()
    </div>
</div>

```

```

<style>
    .e-multi-select-wrapper .e-chips {
        opacity: 0.9;
    }
    .e-multi-select-wrapper .e-chips:hover {
        opacity: 1;
    }
    .e-multi-select-wrapper .e-chips .e-chips-close.e-icon::before,
    .e-multi-select-wrapper .e-chips .e-chipcontent,
    .e-multi-select-wrapper .e-chips .e-chipcontent:hover {
        color: #ffffff;
    }
    .e-chips.chocolate,
    .e-chips.chocolate:hover {
        background-color: #75523C;
    }
    .e-chips.darkorange,
    .e-chips.darkorange:hover {
        background-color: #FF843D;
    }
    .e-chips.darkred,
    .e-chips.darkred:hover {
        background-color: #CA3832;
    }
    .e-chips.fuchsia,
    .e-chips.fuchsia:hover {
        background-color: #D44FA3;
    }
    .e-chips.cadetblue,
    .e-chips.cadetblue:hover {
        background-color: #3B8289;
    }
    .e-chips.hotpink,
    .e-chips.hotpink:hover {
        background-color: #F23F82;
    }
    .e-chips.indigo,
    .e-chips.indigo:hover {
        background-color: #2F5D81;
    }
    .e-chips.limegreen,
    .e-chips.limegreen:hover {
        background-color: #4CD242;
    }
    .e-chips.orangered,
    .e-chips.orangered:hover {
        background-color: #FE2A00;
    }
    .e-chips.tomato,
    .e-chips.tomato:hover {
        background-color: #FF745C;
    }
</style>
<script type="text/javascript">
    function onTagging(e) {
        var colors = document.getElementById('chip-
customization').ej2_instances[0];

```

```
e.setClass((e.itemData)[colors.fields.text].toLowerCase());
}
</script>
```

GAMELIST.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class ColorsData
    {
        public string Color { get; set; }
        public string Code { get; set; }
        public List<ColorsData> GetColorsData()
        {
            List<ColorsData> color = new List<ColorsData>();
            color.Add(new ColorsData { Color = "Chocolate", Code = "#75523C"
});
            color.Add(new ColorsData { Color = "CadetBlue", Code = "#3B8289"
});
            color.Add(new ColorsData { Color = "DarkOrange", Code =
"#FF843D" });
            color.Add(new ColorsData { Color = "DarkRed", Code = "#CA3832"
});
            color.Add(new ColorsData { Color = "Fuchsia", Code = "#D44FA3"
});
            color.Add(new ColorsData { Color = "HotPink", Code = "#F23F82"
});
            color.Add(new ColorsData { Color = "Indigo", Code = "#2F5D81"
});
            color.Add(new ColorsData { Color = "LimeGreen", Code = "#4CD242"
});
            color.Add(new ColorsData { Color = "OrangeRed", Code = "#FE2A00"
});
            color.Add(new ColorsData { Color = "Tomato", Code = "#FF745C"
});
            return color;
        }
    }
}
```

Localization in MultiSelct Control

The Localization library allows you to localize static text content of the [noRecordsTemplate](#) and [actionFailureTemplate](#) properties according to the culture currently assigned to the MultiSelect.

| Locale key | en-US (default)

|-----|-----

| noRecordsTemplate | No records found

| actionFailureTemplate | The request failed

Loading translations

To load translation object to your application, use load function of the **L10n** class.

In the following sample, French culture is set to the MultiSelect and no data is loaded. Hence, the [noRecordsTemplate](#) property displays its text in French culture initially, and if the sample is run offline, the [actionFailureTemplate](#) property displays its text appropriately.

CSHTML

```
@Html.EJS().MultiSelect("customers").Placeholder("Select a
customer").PopupHeight("200px").Locale("fr-BE").DataSource(dataManger =>

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/").Ada
ptor("ODataV4Adaptor").CrossDomain(true)).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings
    {
        Value = "ContactName"
    }).Query("new
ej.data.Query().from('Customers').select(['ContactName',
'CustomerID']).take(0)").Render()
<script>
    ej.base.L10n.load({
        'fr-BE': {
            'multi-select': {
                'noRecordsTemplate': "Aucun enregistrement trouvé",
                'actionFailureTemplate': "Modèle d'échec d'action"
            }
        }
    });
</script>
```

LOCALIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult localization()
        {
            return View();
        }
    }
}
```

See Also

- [Accessibility](#)
- [How to bind the data to the combobox](#)

CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the background color of wrapper element

Use the following CSS to customize the background color of wrapper element.

```
`css
.e-multiselect.e-input-group .e-multi-select-wrapper {
background-color: red;
}
`
```

Customizing the appearance of the delimiter wrapper element

Use the following CSS to customize the appearance of delimiter wrapper element.

```
`css
.e-multiselect .e-delim-values {
-webkit-text-fill-color: blue;
font-size: 16px;
font-family: cursive;
}
`
```

Customizing the appearance of chips

Use the following CSS to customize the appearance of selected chips.

```
`css
.e-multiselect .e-multi-select-wrapper .e-chips .e-chipcontent {
font-family: cursive;
font-size: 20px;
-webkit-text-fill-color: blue;
}
.e-multi-select-wrapper .e-chips {
background-color: aqua;
height: 26px;
}
`
```

Customizing the dropdown icon's color

Use the following CSS to customize the dropdown icon's color.

```
`css
```

```
.e-multiselect.e-input-group .e-input-group-icon, .e-multiselect.e-input-group.e-control-wrapper .e-input-group-icon:hover {  
  color: red;  
  font-size: 14px;  
}  
`css
```

Customizing the focus color

Use the following CSS to customize the focusing color of input element.

```
`css  
.e-multiselect.e-input-group.e-control-wrapper.e-input-focus::before, .e-multiselect.e-input-group.e-control-wrapper.e-input-focus::after {  
  background: #c000ff;  
}  
`css
```

Customizing the disabled component's text color

Use the following CSS to customize the text color when the component is disabled.

```
`css  
.e-multiselect.e-disabled .e-multi-select-wrapper .e-delim-values {  
  -webkit-text-fill-color: red;  
}  
`css
```

Customizing the color of the placeholder text

Use the following CSS to customize the text color of placeholder.

```
`css  
.e-multiselect input.e-dropdownbase::placeholder {  
  color: red;  
}  
`css
```

Customizing the placeholder to add mandatory indicator(*)

Use the following CSS to add the mandatory indicator * to the float label element.

```
`css  
.e-input-group.e-control-wrapper.e-float-input .e-float-text::after {  
  content: "*";  
  color: red;  
}  
`css
```


`

Customizing the float label element's focusing color

Use the following CSS to customize the focusing color of float label element.

```
`css
.e-float-input.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-control-
wrapper.e-input-group:not(.e-float-icon-left) .e-float-line::before,.e-float-input.e-input-group:not(.e-
float-icon-left) .e-float-line::after,.e-float-input.e-control-wrapper.e-input-group:not(.e-float-icon-left)
.e-float-line::after {
background-color: #2319b8;
}

.e-multiselect.e-input-group.e-control-wrapper.e-float-input.e-input-focus .e-float-text.e-label-top, .e-
float-input.e-control-wrapper:not(.e-error).e-input-focus input ~ label.e-float-text {
color: #2319b8;
}
```

`

Customizing the outline theme's focus color

Use the following CSS to customize the focusing color of outline theme.

```
`css
.e-outline.e-input-group.e-input-focus:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-
disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-input-focus.e-control-wrapper:hover:not(.e-
success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left),.e-outline.e-input-group.e-
input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled),.e-outline.e-input-group.e-
control-wrapper.e-input-focus:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled) {
border-color: #b1bd15;
box-shadow: inset 1px 1px #b1bd15, inset -1px 0 #b1bd15, inset 0 -1px #b1bd15;
}
```

`

Customizing the background color of focus, hover, and active item's

Use the following CSS to customize the background color of focus, hover and active item's.

```
`css
.e-dropdownbase .e-list-item.e-item-focus, .e-dropdownbase .e-list-item.e-active, .e-dropdownbase .e-
list-item.e-active.e-hover, .e-dropdownbase .e-list-item.e-hover {
background-color: #1f9c99;
color: #2319b8;
}
```

`

Customizing the appearance of pop-up element

Use the following CSS to customize the appearance of popup element.

```
`css
.e-dropdownbase .e-list-item, .e-dropdownbase .e-list-item.e-item-focus {
background-color: #29c2b8;
color: #207cd9;
font-family: emoji;
min-height: 29px
}
`
```

Customizing the color of the checkbox

Use the following CSS to customize the color of checkbox.

```
`css
.e-popup .e-checkbox-wrapper .e-frame.e-check, .e-popup .e-checkbox-wrapper:hover .e-frame.e-check
{
background-color: green;
color: white;
}
`
```

Accessibility

The MultiSelect control has been designed, keeping in mind the **WAI-ARIA** specifications, and applies the WAI-ARIA roles, states, and properties along with **keyboard support**. This control is characterized by complete keyboard interaction support and ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The MultiSelect component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the MultiSelect component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) | |

| [Section 508 Support](#) | |

| [Screen Reader Support](#) | |

```

| Right-To-Left Support |  |
| Color Contrast |  |
| Mobile Device Support |  |
| Keyboard Navigation Support |  |
| Accessibility Checker Validation |  |
| Axe-core Accessibility Validation |  |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The MultiSelect control uses the **Listbox** role, and each list item has an **option** role. The following **ARIA attributes** denote the MultiSelect state.

| Properties | Functionalities |

| --- | --- |

| aria-haspopup | Indicates whether the MultiSelect input element has a popup list or not. |

| aria-expanded | Indicates whether the popup list has expanded or not. |

| aria-selected | Indicates the selected option. |

| aria-readonly | Indicates the readonly state of the MultiSelect element. |

| aria-disabled | Indicates whether the MultiSelect control is in a disabled state or not. |

| aria-activedescendent | This attribute holds the ID of the active list item to focus its descendant child element. |

| aria-owns | This attribute contains the ID of the popup list to indicate popup as a child element. |

Keyboard interaction

You can use the following key shortcuts to access the MultiSelect without interruptions.

| Keyboard shortcuts | Actions |

| --- | --- |

| Arrow Down | Set focus at the first item in the MultiSelect when no item selected. Otherwise, moves focus next to the currently selected item. |

| Arrow Up | Moves focus previous to the currently selected one. |

| Page Down | Scrolls down to the next page and set focus to the first item when popup list opens. |

| Page Up | Scrolls up to the previous page and set focus to the first item when popup list opens. |

| Enter | Selects the focused item, and popup list closes when it is in open state. |

| Tab | Focuses on the next TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the control. |

| Shift + tab | Focuses on the previous TabIndex element on the page when the popup is closed. Otherwise, closes the popup list and remains the focus of the control. |

| Alt + Down | Opens the popup list. |

| Alt + Up | Closes the popup list. |

| Esc(Escape) | Closes the popup list when it is in an open state and the currently selected item remains the same. |

| Home | set focus to the first item. |

| End | set focus to the last item. |

Note: In the below sample, focus the MultiSelect control using alt+t keys.

CSHTML

```
@Html.EJS().MultiSelect("games").Placeholder("Select a
game").PopupHeight("200px").DataSource((IEnumerable<object>)ViewBag.data).Re
nder()
<script>
    document.onkeyup = function (e) {
        if (e.altKey && e.keyCode === 84 /* t */) {
            var dropObj = document.getElementById("games"); //to get
            dropdown list object
            dropObj.ej2_instances[0].focusIn(); // call the focusIn method
            to focus the element.
        }
    };
</script>a
```

ACCESSIBILITY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult accessibility()
        {
            ViewBag.data = new string[] { "Badminton", "Basketball",
            "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
            return View();
        }
    }
}

```

Ensuring accessibility

The MultiSelect component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the MultiSelect component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the MultiSelect component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

How To

Show the list items with icons

You can render **icons** to the list items by mapping the **iconCss** field. This **iconCss** field create a span in the list item with mapped class name to allow styling as per your need.

In the following sample, icon classes are mapped with **iconCss** field.

CSHTML

```

@Html.EJS().MultiSelect("icons").Placeholder("Select a social
media").PopupHeight("200px").DataSource((IEnumerable<object>
)ViewBag.icondata).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Value =
"SocialMediaName", IconCss = "Class" }).Render()
<style>
    @@font-face {
        font-family: 'Socialicons';
        src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMvltCfsAAAEoAAAAVmNtYXCnKKeOAAABrAAAAEhnbHlml19
XagAAAgwAABhQaGVhZA8dCeEAAADQAAAAANmhoZWEIUQQMAAAARAAAACRobXR4LAAAAAAAYAAAAA
sbG9jYR3AIwWAAAH0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGX5D00
AABY0AAAAkwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAACwABAAAAAQAA+iTiP18
PPPUACwQAAAAAANYFYngAAAAAlgVieAAAAAAD9AP0AAAACAACAAAAAAAAAAAAAEAAAAALafQACwAAAAA

```

AAgAAAAoACgAAAP8AAAAAAAAAAQQA ZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQM AAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnQQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQA AwABAAAFAA
EADQAAAAEAAQAAQAApwn//wAAPwD//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAiQ
CzgOMBU4F/gZYB9QICaO+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fHQEfDhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLWYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kiw8FAYkFEgQDAyQDAQECAyIBAQMSegkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgWKCQQEBQICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwCFBQUEAwICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRUVFBU
VDw4ODg0NDawLCwoKCQkICaCGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwWNDQ0ODg4PDw8QEBA
QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwWNDQ4ODxAREhISEhMTExMUExQUFRQ
VGXsaGgcIBwFXNgEBAQ8KCgoIBwCGBQUDAwIBAQECAwMDBQUBGcHCAgICgkKCwsMDAwNDg0ODw8
PEBAQEhISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCACh/aAQEB0cGhgWFBIRDgw
LCaCEAwICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQEASODhAQDQ0LCQg
HBgQCAgICBAQEBgYGCAGJCQoLCwWMDQ4ODg8PEBAREBESEhISExITGBcZGRgZGBgXAU4CagMEXAk
FBAQFBCQCAwMGHcKFQkMIwIBAwYAwIBKQECBSgFBgULCgkHBgMBAQEDAwICGwMDg8PEhITFBU
WGBgSEyYJCAgIBwCHDBEGAgEBAagEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhE
SEREREBEPEA8QDhMEBASFNgEBAQEJCAcGBQMCAQEDAwUGCAgHCAgJCQoKCwsMDA0NDQ4ODg8ODxA
PEA8QEBAQEBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICaCGBgUFAwMDAQEBAQEC6RISExISExISEhM
SEhESERERERAQEASQDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVncFBAQVEBEREhESEhI
TExMTFBMUEhEREBEQEBA PDw8PDg4ODQ0MDAwLCwoKCgkICaCHBwUGBQQDAgIBAQEBAgIDBAQFBQc
GCAGICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUfHcYGRScHR8fIiIjFBMTEhMSEhIREhEREBEQEAM
EAWt1YQINCAYEAgIEBAUGBgICQoKDAwNDg8PERUVFhcWGBcYGBkZGRkaGxoTEhISEhEREBAQDw8
ODg4MDQwLCgoKCQgHBWYFBQMDAwEBAGMFBQgIAAAAAEAAAAAAzoD9ACWAAATDwYVERUfHTsBPw4
9AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwWMDA0NDQ0ODg4PDw8
QEBDQCGsKCQkJCAGHBWUEBAICAgIEBAUHBwgICQkJCgsK0QoKCgoJCAgIBwCFBAMDAQEBKQoJCgg
JCAgHBwUFBAQCAQEBAQIEBAUFBwgHCAkJCQkK/tcCagMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgs
KDAwMDf4LExMTEhESEREQEBA PDw4PDQ4MDAOKCQgIBgYEBAMCAgEBAwQFBwJCQoKCwsMDA0NDAs
MCwoKCQkHBWUEAwEBAQEDBAUGCAgJCgoLCwWMDVgCAwMFBGcICQkJCgsLCwWLDAsKCgoJCAgHBgU
EAgIBtQ0MDAsLCgoJCQcHBQQAQEBAQMEBQYIAAABAAAAAAP0A90AqAAAA8DMx8MHQEPDSsBLxo
PCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBWUFbQQA DAwICAgEDBQoPEXyWFAsLCgQFBAUFBAU
FBQUJCQkJCYQEBAQUGBwcICAKKCsLDQWODQ8RERQfI5IvNRMGBWYGBWYGBgYGDAsMWAcICAgICQk
JCQkKCgoLCgsJEXITFBQVFRYWFIMkJTEWFRQREQ8NDAsJBWYFAwEBAQEDBAUHBwkJCwWNDhAQEGs
YGBYWFbQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBQUHBGcICBMSDxEdIiUoIXsOCgcCAQIEBAYICBU
bHyU37xQTERAPDQWKCQgGBQMCAQEDBgLDhofkUInCwIBAgQEBwJCwscISf+pBYUEXIQDg4LCwk
IBgUDAgEDBACJDA0REhQXJysxRiEhIB4eHRwbGhkZFxcVFRoXfHqTERAODQWKCACGBAMBAQMFBwk
MDQ8REXUXGhsdAAUAAAAA/ED9ABCAKoA6wESAYQAAEdAQ8NKwEvDjU/EB8OJR0Bhw8hPw8TLwM
hHwUVDxEvEzU/CSchDwMFFR8PPw8vDw8OAR8HFQ8JiY8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM
9Ai8BIw8EFQ8DIY8CNw8KFxUfASUzPwgZHWkhPwI1LxAlDwICkAMDBQcHCQkKDAwMDQ4ODwWMCws
LCgoJDwsJCAyFAgECAwQGBgcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBGkJDAwODwg
RERITewJpExMSEhEQDw4MDAUJBWYEAgEBAgEF/uYOCwKGBAICBAYHCQsMDg8QERETExQTFRQVFBQ
UFBMTEhAPDg0LCQgGBAMCAgEBAwMEBQYHCAKc/uoFAgEBASwBAWUGCQoLDQ0PERESEhQUFBQTEhE
QEA4MDAKJBgUDAQEDBQcICgsNDg8QERISFBQUFBMSERAQDgWMCgghBAMCPAYGBgQEAWEBABQEBAM
EBAQFBmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFeAKDAwECBAQEBAQDAWI
CAWUWAwIBAQQDdwLCQgFAwEBAQEEATEEBBYUFRYWFxcYFxcXFxYVFBgFBQYBJwYCAgICBAYHCQo
LDA4ODxAIERIR/d8FAQIB9wcHDg0NDawLCgkIBwYFBAICAgQEBQYGDQwODg8REBINDQwMCwsKCgk
IBwCGBAQCAQEBAgQFBggICgoLDQ0NDg9929sUEXISERAPDgWMBQkHBgQCAQMFBGkJDAwODwgQERI
TEwHBBgMBARYXFxcXFxcWFhUUFBMREQ8ODAsJCAYEAWIBAgQFBWkKDA0PDxEREXQPEA8PDw8PDw8
ODw4ODg4OAQEBAQKPCgoUEhIREBANDQsKCAcFAwEBAUHCAoLDQ4PEBESExQUFBMTEhEQDw4NCwo
IBwUDAQEDBQcICgsNDg8QEHITewGSAQICBAUFBgdsBQQFBAQDAgIBAQECAgQFBQYHawCHBgUDAgE
BR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDA WMG0ggEAQICAgTKAQ00EBASEhQ
VEiRdAgIBAQITDg0JCAyDAQQFBwoMDhQCAQEBAQNuJBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAM
AAAAA/QD3QADAFcAlwAANzMRIwUVIzc1IXeZET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQH
W1gIDBQgKCwCHBwgJCQoKCw4NDAsKCAgHBWUEBAICAQHWAQICAgQDBQUFBgYHBwCJCAkJCgoKCws
LDBgZGhQUEREPDg0MCwoJCQ79xAEBAwMFBgYHCAKkCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAY
GCAGJCgoLDQWODQ0MDAOKCQkHBWYEBAMBIGKFWwICW/17AXcUDA0ODgWGBQUEBAMCAQEBAgMFBQc
ICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQWMCwoKCQkICaCGBgUFBAMGAwEBAgMEBgYHCAgICQkSARI

MCwSKCgkICAgGBQUEAwEBAQEDBAUFBgICAKKCgsLDAsLCwsJCggIBwYGBAQDAQEBAQMEBAYBwG
ICgkLCwsAAAAABAAAAAPuA/QARgAAExEVHwYhESM1MzU/DzMVIw8GFTMVIxehPwYRLwYhDwYSaGQ
FBwGKCgHPb24BAwMGBggJCgsMDQ00DwgPlUcLCwkIBgQDe3sBBQoKCAcFBAICBAUHCAoK/IUKCgk
HBwQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBwUEAgGFawQHCAkKDDOF/mUDBAYICQoLA4I
LCgkIBgQDAQQFBwGKCwAAAAAGAAAAAAP0A/QAOABEAiABBQEqAUwAAAEPCROBHw07AT8NPQEVC
PASUVMxUjFSM1IzUzNSUPBRUfDTsBPwW1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIHIy8NPQE
/DwEVHw8hPw8RITChLw8hDw4BCgMTCwsFBAQEAgICAwQGBGcICGoLDAwODg8NDQwLCgkICAYGBQ
DAwEBAQIDBAgMDiYRNw0B9nR0TXNz/kAFawMDAQIBAgMDBAQGBGcICQkKDasIBwCHBwYFBQYFAwM
BAQECAwMEBQYGBwGJCQoLDAcIBwCHBwX+MTAQDggIAwICAQEBAQEDAwMICGwsMDAsGAgEBAQECawY
iGQoFCQcDagIBAwQFCAGLDA0PERITFRYFRISEA8NDAsKCAcGBAMCAQEDAwUHQCsOERMUFBOxCA
DAWEBAQIFGQ40DQ0LCgoICAcFBQQCAgMDBGcICGwICBESEhESEBD+pwEDBQYJCgsNDg8IEBISExQ
CahQTExIREA8ODQsGCQcGBAL8GAED5gIDBgICGsnDg4QCBIRExp9lhMTExERE40DQsKCAcGaw
KAQkHCAYGBGgICQkKCgkICAgHBgYFBQMDAgIBAgMDBAUFBgYHBwCICQgHBwYGBgYLCwwcBQPYck9
yck5zZwYGBwCHDxELCgWLCwsKCgkJBwUFAwEACawMDBAUHBwCIBw0QCwwLDAsMCgoKCAcGBAMBAgI
DAwQFLRkQDwwPCAgJCgoLCQkICAgNDAsKCQwJBQYGBQYEBACbFQsGDA4HCAgJCQ4NDQwNCwwKCg
IBgYDAwEBAgMDBQYGBwGJCAoJCgsKCwUMDAwMDAsKCQYFBQUKDAYHCAgJBw0BAgQEBQcHCAkJCgo
KCwsLDQ4NDQ0MDAsGBgkIBQQCAQH+EAoKEXMSERAQDQ0LBgkHBGQCAQMFBGkKCw0NEAgQEHITFAJ
HKxQSEhIQDw8NDAsJBQcFBAIBAwQHCAkLDA0PDxASEhIAAAAAAGAAAAAD7gP0AEAAhAAAAARUzFSM
RHws/BxUPAy8OESM1Pw81ER8OMyEzPw4RLw4jISMPDQIBysODBgUICgYHCAgJCgsLDQ4PEBESE0Q
tICIiEREQDw8ODQwKCgCHBANuGBkVDw4ODgYFBgUEBAMCAv5fAQECawQEBQUGBwCHCAgJCAM0CAk
ICACHBwYFBQQEAWIBAQEBAgMEBAUFBgCHBwGICQj8zAgJCAgHBwCGBQUEBAMCAQON0H/+9BIMCAk
HBAMDAgEBAQEBAgMDBQYHeA4GAwEBAgIDBAUFBwGJCwsNDxABVGwKDXANDxEUCwwMDQ00DxAQEvz
CCQgICACHBwYGBAUDAwICAgIDAwUEBgYHBwCICAgJaz4JCAgIBwCHBgYEBQMDAgICAgMDBQQGBGc
HBwGICAAAAAGAAAAAD7APzAPgBqAAAAAR8LFQ8MIy8QKwEPdH8bHQEPfi8WPQE/DTMfEjM/Di8ePQE
/Fh8C8R8HDwMfHjSBPwIFBzM/HTUvBz8CPQEvHiMpaI8HIw8dAnALFhMSDw4LCQgFBAIBAgIDAw
FBGUGBgCGCAwLCQgHChQLCwsHBwKJCgsNDQgMCwsJCggIBwYFBAMDAQEBAGMEBQcHCBMTDxojfHq
TEA8OCwUFAwQCAwEBAgIEBQUHCAgKCgWMDg4PEBEREHMTFBUZGBYWFRMSEgSLCwoJCQgIBwYFBQM
CAgECAgMDBAUFBQYGBgYHCAsLCgkIBwCMBwCHBwKDAcPERMZDQ0MDAsKCQgHBGUEAwEBAQICAgM
EBASMDQ8bTSIfGxkMCwsKCQgIBwYFBQMCAgICBAQGBGgICQoLDA0NDw8PEREREXIUHxwb/bsBAgM
EBQcHCQUADAQEBAQMFBQYICAKLCwWNDg8QEBESEhMUFBWFRcWGBcYGBYWFRUPDXAQEBEREQ40Dg0
NDQ0MDAwMCwoLCgkJCQgHBwCGBGQFAwMDAgEBAQIDBAUGBgQEAgIDBAUHBwKJCgWMDQ4PDxERERM
TFBQVFRYWFxcYGBGUFQRTEBESEhITFBMODg4NDQ0NDA0LDAsKCwoJCQkIBwCHBgYEBQMDAwIBazc
ECAoLDAwNDQ4NDg0NBgYGBQYKBQDQAwICAQECAUHDSEODQoEBAMCAgIBAQCICAwMEBQUFBQYGBgY
GCAChBgYFBQUIBx0GDAgJCgsNDg8JCAkKCgsLCwwPDg0ODQwMDAsLCgoICAgHBgUEBAMCAQEBAgI
EBQYICAYIBwKJCQoKCwsLCgsLCgoHBgYGBQUFBQQEAWMCAQEBAgUGCAkLGg0LCgkICAYDBAMCAQI
DBAQFBgYGBwCIBwKIDQcFBgUEBQgIBgYHEgkJCgoHBGcICAKJCgoLDAwMDg0NDQ0MDAsLCgoKCQg
IBwYGBQQEAWMBAQEBAwRbEhMSEREREBAXFxcYGBGyYFxcWFhUvFBQTEEXEREQ8PDg0MDAoKCAChBQ
DAGICAwCGBGUDAwEBAQIDAwMFBAyGBwCHCAkJFCoLcGSMDAwMDQ0NDQ40DhAQEA8PDw4OGB0ZGhg
YFhgWfXUWFRQUExISERAQDw4NDAsLCQkHBGUFawEBAgIDCgYHBGUDAgEBAgMDAwUEBgYGCAcICQK
JCgoLCwwLDQwNDQ0NDg4AAAAACwAAAAAD8wOYABEAMwBbAKyAywDtarCBOQfjAzgBoQAAaQMDzc
vBisBDwEnDwIdAh8FOWE/BjUvBisBDwEnFwcfBDM/Bic1MxUnNw8GIy8HNyClHwsVixUfBjsBPwY
1MxcVDw0vCzU/CycVPwMfCR0CDwgjLwQPAREjFSMVIzUjNtPCxUfDyE/DzUvDiMhIw8BJR8DFQ8
GKwEvBjU/Bx8DFR8KPwUHMzUjFQ8GKwEvBjUjDwcdAR8LowE/CTUvDg8DFTM1NyMHJyMDIgQDAgJ
CAgECAwQFBQYGCgUG2QQDAgIDBAUFBQYGBQYEBQICAECAgUEBgUGBgUF6wEBAwUCAwMEBwGEGAgE
BAQFFOQEDBAYMDhAQDwgGBgYFAgIEAQECHQoLCgkJCAcFBAMCAxCAQMDBQQFBhAGBQQEAgIBMwM
BAQMCBAQMBwCICAgIDxAPDg8HBgYEAwMBAQECAgQGBGgJCQkJC+UQDg4NDQUGBwCfBAQCAgIDAwU
GBGcICQoLBQwNFAQ50VJFRyAPDQ0LCwkJBwMFAwIBAgQGBwKJCwsNDQ8PDxARAQAEQ8PDw0NCws
KCAcDBQMCAQIEBgCICgsLDQ0PDxAQEP1gERAPAYoEAgIBAQCICBAUGBQYGBGUFBAICAECAwQFBQU
GBGUGdgEEAgQEBQQGBwGIBwCGBGoKAUw7AQEDAwQEBQUFBQQEAWIBAUC+CggGBGMCAGICBAMECgY
HCQsLCwwMCwoSBwCHCAUEAgEBAgIEAwQHBwGJCgsMDg8NDMPKV1AuLVEBbAMEBB8bBQMEAwICAQE
CCAMDAwOAAwMDAgICAQECAgIDAwOAAwMDAwICAQECF4kgBQUCAQECAQEAgIDCJrYARsEBAMHBQQ
CAQICAwQEBRoPmw0BAQIDAwQFBgYMDBgvMgYEAwIDAQEBAQMCAwQEGREGBgUFBQQECQQEAWICAQE
BAwYHBQYGBwGJCgPDDwwLCgGHBgYDAwMBQFQHBQQBAQICAwUFBQYHBwhwCgkJCAgGBQDQAEQBBAU
LEgEBIib+/yVJBQUGBwCICQkFCgsK9gsLCgoJCQgHBwYFBQMDAgEBAQMDBQUGBwCICQkFCgoL9wo
LCgoJCQgHBwYFBQMDAgID9wQEBABV3BAUDBAMCAQECAwQEBAR3BQQEBAwQEBQJ3GAWRBAUEAwM
DAQEBAQEBAwCKEuCvAwMDAgMBAQEBAwIDAwOvAgUHCAoKDA0QXgOBwCGCgQEAGMCAQICBQQQFBws
KDxZTCgkHBGyYGBQYFBQMDAgEBAQIEPayslG9vAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAA

```
AAQABAAAAAAACAAcADAABAAAAAADAAAsAEwABAAAAAAAEAAAsAHgABAAAAAAAFAAAsAKQABAAAAAA
GAAsANAABAAAAAAAKACwAPwABAAAAAALABIAawADAAEECQAAAAIAfQADAAEECQABABYafwADAAE
ECQACAA4AlQADAAEECQADABYAowADAAEECQAEABYAuQADAAEECQAFABYAzWADAAEECQAGABYA5QA
DAAEECQAKAFgA+wADAAEECQALACQBUyBTb2NpYWxpY29uc1JlZ3VsYXJTB2NpYWxpY29uc1NvY2l
hbG1jb25zVmVyc2lvbiAxLjBTb2NpYWxpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAUwBvAGMAaQBhAGwAaQBjAG8AbgB
zAFIAZQBnAHUAbABhAHIAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAFMAbwBjAGkAYQBsAGkAYwBvAG4
AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAUwBvAGMAaQBhAGwAaQBjAG8AbgBzAEYAbwBuAHQAIA
nAGUAbgB1AHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIABNAGU
AdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQA
AAAAACAAAAAAAAoAAAAAABAAAAAABAAAAAABAAAAAABAgEDAQQBBQEGAQcBCAEJAQoBCwE
MAAh3aGF0c2FwcAd0d2l0dGVyBXZpbWVvCWluc3RhZ3JhbQhsaW5rZWRpbgghmYWNlYm9vawtnb29
nbGUTcGxlcwZ0dWlibHIIc2t5cGUtMDEIeW9ldHViZTEAAAA=) format('truetype');
```

```
font-weight: normal;
```

```
font-style: normal;
```

```
}
```

```
.e-list-icon {
```

```
font-family: 'Socialicons' !important;
```

```
color: rgba(0, 0, 0, .57);
```

```
}
```

```
.twitter:before {
```

```
content: "\a701";
```

```
}
```

```
.vimeo:before {
```

```
content: "\a702";
```

```
}
```

```
.youtube:before {
```

```
content: "\a709";
```

```
}
```

```
.whatsapp:before {
```

```
content: "\a700";
```

```
}
```

```
.skype:before {
```

```
content: "\a708";
```

```
}
```

```
.instagram:before {
```

```
content: "\a703";
```

```
}
```

```
.google-plus:before {
```

```
content: "\a706";
```

```
}
```

```
.facebook:before {
```

```
content: "\a705";
```

```
}
```

```
.tumblr:before {
```

```
content: "\a707";
```

```
}
```

```
.linkedin:before {
```

```
content: "\a704";
```

```
}
```

```
</style>
```

ICONS.CS

```
using System;
using System.Collections.Generic;
```



```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class MultiSelectController : Controller
    {
        public ActionResult icons()
        {
            ViewBag.icondata = new SocialMedia().SocialMediaList();
            return View();
        }
    }
}

```

Set preselected items through fields

You can use a boolean field(for ex:"isSelected") of MultiSelect dataSource to set preselected items through fields during initial rendering. You can use `itemCreated` event of fields to push items with **isSelected** field set to true and these values will be selected through [dataBound](#) event of MultiSelect. Pass empty string of array initially to load the control with preselected values.

In the following sample, selected values are mapped through **isSelected** field.

CSHTML

```

<div id='iconList' class='col-lg-6' style='padding-top:15px'>
    <div class='content'>

@Html.EJS().MultiSelect("local").Width("300px").Value(ViewBag.val).Placeholder(
"Select countries").Mode(Syncfusion.EJ2.DropDowns.VisualMode.CheckBox).DataSource((I
Enumerable<Object>)ViewBag.data).ActionBegin("onBegin").DataBound("onBound")
.Render()
        <script>
            var selected = [];
            function onBegin(e) {
                this.fields = {
                    text: 'Name', value: 'Code', itemCreated: function (e) {
                        var count = 0;
                        if (count === 0) {
                            for (let i = 0; i < e.dataSource.length; i++) {
                                if (e.curData.isSelected == true)
                                    itemSearch(e.curData.Code); //pass the
corresponding value
                            }
                        }
                    }
                }
            }
            function itemSearch(e) {
                if (selected.indexOf(e) == -1)
                    selected.push(e);
            }
            function onBound(e) {
                this.value = selected;
            }
        </script>
    </div>

```

```

    }
    </script>
</div>
</div>

```

COUNTRY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
    public class Countries
    {
        public string Code { get; set; }
        public string Id { get; set; }
        public string Name { get; set; }
        public bool isSelected { get; set; }
        public List<Countries> CountryList()
        {
            List<Countries> country = new List<Countries>();
            country.Add(new Countries { Id = "ct1", Name = "South Africa",
Code= "001" ,isSelected=true });
            country.Add(new Countries { Id = "ct2", Name = "North America",
Code="002", isSelected = false });
            country.Add(new Countries { Id = "ct3",Name = "West
Indies",Code="003", isSelected = true });
            country.Add(new Countries { Id = "ct4", Name = "North
California" ,Code="004", isSelected = true });
            country.Add(new Countries { Id = "ct5", Name = "East
Indies",Code="005", isSelected = false });
            country.Add(new Countries { Id = "ct6", Name = "India", Code =
"006", isSelected = false });
            country.Add(new Countries { Id = "ct7", Name = "Japan", Code =
"007", isSelected = true });
            country.Add(new Countries { Id = "ct8", Name = "China", Code =
"008", isSelected = false });
            return country;
        }
    }
}

```

MultiSelectFor

The MultiSelectFor control can be rendered by passing values and data from the model. The selected values can be retrieved during form submit using the post method.

In the following sample, MultiSelectFor control is rendered.

CSHTML

```

@model Mvc_test.Models.Countries
@using (Html.BeginForm())
{
    @Html.EJS().MultiSelectFor(model => model.Values).Placeholder("Select a
Country").DataSource((IEnumerable<object>) ViewBag.data).Fields(new

```

```

Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Value = "Name"
}).Render()
<div id="submitButton">
    @Html.EJS().Button("btn").Content("Post").Render()
</div>
}

```

FOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        Countries model = new Countries();
        public ActionResult Index()
        {
            ViewBag.data = new Countries().CountriesList();
            model.Values = new string[] { "Cameroon" };
            return View(model);
        }
        [HttpPost]
        public ActionResult Index(Countries model)
        {
            ViewBag.data = new Countries().CountriesList();
            model.Values = model.Values;
            return View(model);
        }
    }
}

```

Data Annotation

Data Annotations help us to define the rules to the model classes or properties for data validation and displaying suitable messages to end users.

Data Annotations includes built-in validation attributes for different validation rules, which can be applied to the properties of model class. ASP.NET Framework will automatically enforce these validation rules and display validation messages in the view.

Using **value** property gets or sets the value of the selected item in the control.

CSHTML

```

@using (Html.BeginForm())
{
    @Html.EJS().MultiSelectFor(model =>
model.EnquiringAboutSelect).Placeholder("Select a
Country").DataSource(Model?.EnquiringAboutSelectListItems).Fields(new
Syncfusion.EJ2.DropDowns.MultiSelectFieldSettings { Value = "Text"
}).Placeholder("Please Select Enquiring About").Render()
<div>

```

```

        @Html.ValidationMessageFor(model => model.EnquiringAboutSelect)
    </div>
    <div id="submitButton">
        @Html.EJS().Button("btn").Content("Post").Render()
    </div>
}

```

FOR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
    public class AutoCompleteController : Controller
    {
        public ActionResult Index()
        {
            DataModel model = new DataModel();
            model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
            model.EnquiringAboutSelect = new string[] { "104" };
            return View(model);
        }
        [HttpPost]
        public ActionResult Index(DataModel model)
        {
            model.EnquiringAboutSelectListItems = new
ListItems().getListItems();
            model.EnquiringAboutSelect = model.EnquiringAboutSelect;
            return View(model);
        }
    }
    public class DataModel
    {
        [Required(ErrorMessage = "The value is Required")]
        public string EnquiringAboutSelect { get; set; }
        public List<ListItems> EnquiringAboutSelectListItems { get; set; }
    }
    public class ListItems
    {
        public string Text { get; set; }
        public string Value { get; set; }
        public List<ListItems> getListItems()
        {
            List<ListItems> items = new List<ListItems>();
            items.Add(new ListItems() { Text = "Aberdeen", Value = "103" });
            items.Add(new ListItems() { Text = "Alexandria", Value = "102"
});
            items.Add(new ListItems() { Text = "Albany", Value = "101" });
            items.Add(new ListItems() { Text = "Beacon ", Value = "104" });
            items.Add(new ListItems() { Text = "Brisbane ", Value = "105"
});
        }
    }
}

```

```

        return items;
    }
}

```

Migration from Essential JS 1

This article describes the API migration process of multiselect component from Essential JS 1 to Essential JS 2.

Accessibility and Localization

Behavior | Property in Essential JS 1 | Property in Essential JS 2 |

|-----|-----|-----|

| Localization | **Property** : Locale @Html.EJ().DropDownList("locale").Locale("de-DE") |

Property : Locale

@Html.EJS().MultiSelect("locale").Locale("de-DE").Render() |

| Right to left | **Property**: EnableRTL @Html.EJ().DropDownList("rtl").EnableRTL(true) |

Property: EnableRtl

@Html.EJS().MultiSelect("rtl").EnableRtl(true).Render() |

Animation

Behavior | Property in Essential JS 1 | Property in Essential JS 2 |

|-----|-----|-----|

| Animation | **Property** : EnableAnimation

@Html.EJ().DropDownList("default").EnableAnimation(true) | Not Applicable |

Template

Behavior | Property in Essential JS 1 | Property in Essential JS 2 |

|-----|-----|-----|

| Header Template | **Property** : HeaderTemplate

@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Employee1>)ViewData["LocalDataSource"]).HeaderTemplate("<div class='eheader'>PHOTODETAILS</div>") | **Property** : HeaderTemplate

@Html.EJS().MultiSelect("remote").HeaderTemplate("NameCity").Render() |

| Item Template | **Property** :ItemTemplate

@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Employee1>)ViewData["LocalDataSource"]).Template("<div><Image class='ImageId' src='../Content/Employees/{Image}.png' alt='employee'/> <div class='ename'> \${Text} </div><div class='role'> \${Role} </div><div class='cont'> \${Country} </div></div>") | **Property** : ItemTemplate

@Html.EJS().MultiSelect("remote").ItemTemplate("<div>\${FirstName}\${City}</div>").Render() |

| Footer Template | **Property** : Not Applicable | **Property** : FooterTemplate

@Html.EJS().MultiSelect("remote").FooterTemplate(" Total list items: " + 4 + "").Render() |

| Group Template | **Property** : Not Applicable | **Property** : GroupTemplate

@Html.EJS().MultiSelect("remote").GroupTemplate("\${City}").Render()
 |

| Value Template | **Property** : Not Applicable | **Property** : ValueTemplate

@Html.EJS().MultiSelect("remote").ValueTemplate("\${FirstName} -
 \${City}").Render() |

| No Records Template | **Property** : Not Applicable | **Property** : NoRecordsTemplate

@Html.EJS().MultiSelect("remote").NoRecordsTemplate(" NO
 DATA AVAILABLE").Render() |

| Action Failure Template | **Property** : Not Applicable | **Property** : actionFailureTemplate

@Html.EJS().MultiSelect("remote").ActionFailureTemplate("
 Data fetch get fails").Render() |

Data Binding

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Data Source | **Property** : Datasource

@Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Employee>)ViewData["LocalDataSource"]).DropDownListFields(Df => Df.Text("Text").Value("Country")) | **Property**
 : Datasource

 @Html.EJS().MultiSelect("default").DataSource((IEnumerable<object>)ViewBag.localdata).Render()
 |

| Query | **Property** : Query

@Html.EJ().DropDownList("customerList").Datasource(ds =>
 ds.URL("https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/").CrossDomain(true)
).Query("ej.Query().from('Customers').take(6)").DropDownListFields(f => f.Text("CustomerID"))
 | **Property** : Query

@Html.EJS().MultiSelect("remote").Query((string)ViewBag.query).DataSource(obj =>
 obj.Url("http://services.odata.org/V4/Northwind/Northwind.svc/").CrossDomain(true).Adaptor
 ("ODataV4Adaptor")).Fields(new MultiSelectFieldSettings { Text = "FirstName", Value =
 "EmployeeID" }).Render() |

| Fields | **Property** : Fields

@Html.EJ().DropDownList("customerList").Datasource(ds =>
 ds.URL("https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/").CrossDomain(true)
).Query("ej.Query().from('Customers').take(6)").DropDownListFields(f => f.Text("CustomerID"))
 | **Property** : Fields

@Html.EJS().MultiSelect("remote").Query((string)ViewBag.query).DataSource(obj =>
 obj.Url("http://services.odata.org/V4/Northwind/Northwind.svc/").CrossDomain(true).Adaptor
 ("ODataV4Adaptor")).Fields(new MultiSelectFieldSettings { Text = "FirstName", Value =
 "EmployeeID" }).Render() |

| Action Begin | **Event** : ActionBegin

 @Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.ActionBegin("onActionBegin")) | **Event** : ActionBegin

@Html.EJS().MultiSelect("list").ActionBegin("onActionBegin").Render() |

| Action Complete | **Event** : ActionComplete

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.ActionComplete("onActionComplete") | **Event** : ActionComplete

@Html.EJS().MultiSelect("list").ActionComplete("onActionComplete").Render() |

| Action Failure | **Event** : ActionFailure

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.ActionFailure("onActionFailure") | **Event** : ActionFailure

@Html.EJS().MultiSelect("list").ActionFailure("onActionFailure").Render() |

| Action Success | **Event** : ActionSuccess

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.ActionSuccess("onActionSuccess") | Not Applicable |

| Data Bound | **Event** : DataBound

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.DataBound("onDataBound") | **Event** : DataBound

@Html.EJS().MultiSelect("list").DataBound("onDataBound").Render() |

Filtering

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Filtering | **Property** : EnableFilterSearch

@Html.EJ().DropDownList("default").EnableFilterSearch(true) | **Property** :
 AllowFiltering

@Html.EJS().MultiSelect("default").AllowFiltering(true).Render() |

| Server Filtering | **Property** : EnableServerFiltering

@Html.EJ().DropDownList("default").EnableServerFiltering(true) | **Event** : Filtering

@Html.EJS().MultiSelect("list").Filtering("onFiltering").Render() |

| Sorting | **Property** : EnableSorting

@Html.EJ().DropDownList("default").EnableSorting(true) | **Property** : SortOrder

@Html.EJS().MultiSelect("default").SortOrder('Ascending').Render() |

| Case Sensitive Search | **Property** : CaseSensitiveSearch

@Html.EJ().DropDownList("default").CaseSensitiveSearch(true) | **Property** : IgnoreCase

@Html.EJS().MultiSelect("default").IgnoreCase(true).Render() |

| Ignore Accent | Not Applicable | **Property** : IgnoreAccent

@Html.EJS().MultiSelect("default").IgnoreAccent(true).Render() |

| Filter Bar Placeholder | Not Applicable | **Property** : FilterBarPlaceholder

@Html.EJS().MultiSelect("default").Mode('CheckBox').FilterBarPlaceholder('search the
 value').Render() |

| Search | **Event** : Search

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e => e.Search("onSearch") |
Event : Filtering

@Html.EJS().MultiSelect("list").Filtering("onFiltering").Render() |

Popups

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

```

|-----|-----|-----|
| Popup Resize | Property : EnablePopupResize<br/>
<br/>@Html.EJ().DropDownList("default").EnablePopupResize(true) | Not Applicable |
| Maximum Popup Height | Property : MaxPopupHeight<br/>
<br/>@Html.EJ().DropDownList("default").MaxPopupHeight(300) | Not Applicable |
| Minimum Popup Height | Property : MinPopupHeight<br/>
<br/>@Html.EJ().DropDownList("default").MinPopupHeight(200) | Not Applicable |
| Maximum Popup Width | Property : MaxPopupWidth<br/>
<br/>@Html.EJ().DropDownList("default").MaxPopupWidth(300) | Not Applicable |
| Minimum Popup Width | Property : MinPopupWidth<br/>
<br/>@Html.EJ().DropDownList("default").MinPopupWidth(100) | Not Applicable |
| Popup Height | Property : PopupHeight<br/>
<br/>@Html.EJ().DropDownList("default").PopupHeight(500) | Property : PopupHeight<br/>
<br/>@Html.EJS().MultiSelect("default").PopupHeight(500).Render() |
| Popup Width | Property : PopupWidth <br/>
<br/>@Html.EJ().DropDownList("default").PopupWidth(300) | Property : popupWidth<br/>
<br/>@Html.EJS().MultiSelect("default").popupWidth(300).Render() |
| Show Popup On Load | Property : showPopupOnLoad <br/>
<br/>@Html.EJ().DropDownList("default").showPopupOnLoad(true) | Not Applicable |
| Close Popup On Select | Not Applicable | Property : ClosePopupOnSelect <br/>
<br/>@Html.EJS().MultiSelect("default").ClosePopupOnSelect(true).Render() |
| Open On Click | Not Applicable | Property : OpenOnClick <br/>
<br/>@Html.EJS().MultiSelect("default").OpenOnClick(true).Render() |
| hidePopup | Method : hidePopup <br/> <br/>@Html.EJ().DropDownList("default")<br/>
<br/>$('#dropdown').ejDropDownList('hidePopup') | Method : hidePopup <br/>
<br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.hidePopup() |
| showPopup | Method : showPopup <br/> <br/>@Html.EJ().DropDownList("default")<br/>
<br/>$('#dropdown').ejDropDownList('showPopup') | Method : showPopup <br/>
<br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.showPopup() |
| Popup Hide | Event : PopupHide<br/>
<br/>@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
e.PopupHide("onPopupHide") | Event : Close<br/>
<br/>@Html.EJS().MultiSelect("list").Close("onClose").Render() |
| Popup Shown | Event : PopupShown<br/>
<br/>@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
e.PopupShown("onPopupShown") | Event : Open<br/>
<br/>@Html.EJS().MultiSelect("list").Open("onOpen").Render() |

```


| Popup Resize | **Event** : PopupResize

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.PopupResize("onPopupResize") | Not Applicable |

| Popup Resize Start | **Event** : PopupResizeStart

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.PopupResizeStart("onPopupResizeStart") | Not Applicable |

| Popup Resize Stop | **Event** : PopupResizeStop

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.PopupResizeStop("onPopupResizeStop") | Not Applicable |

| Before Popup Hide | **Event** : BeforePopupHide

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.BeforePopupHide("onBeforePopupHide") | Not Applicable |

| Before Popup Shown | **Event** : BeforePopupShown

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.BeforePopupShown("onBeforePopupShown") | Not Applicable |

Selection

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Selected Index | **Property** : SelectedIndex

@Html.EJ().DropDownList("default").SelectedIndex(2) | Not Applicable |

| Selected Indices | **Property** : SelectedIndices

@Html.EJ().DropDownList("default").ShowCheckbox(true).SelectedIndices(new List<int> {
 1,2 }) | **Property** : Value

 viewBag.vlue = [1,2]

 @Html.EJS().MultiSelect("default").Value(ViewBag.value).Render() |

| Maximum Selection Length | Not Applicable | **Property** : MaximumSelectionLength

 @Html.EJS().MultiSelect("default").MaximumSelectionLength(5).Render() |

| Select All Text | Not Applicable | **Property** : SelectAllText

 @Html.EJS().MultiSelect("default").SelectAllText('Check All').Render() |

| Un Select All Text | Not Applicable | **Property** : UnSelectAllText

 @Html.EJS().MultiSelect("default").UnSelectAllText('Un Check All').Render() |

| Selection Order | Not Applicable | **Property** : EnableSelectionOrder

 @Html.EJS().MultiSelect("default").EnableSelectionOrder(true).Render() |

| Hide Selected Item | Not Applicable | **Property** : HideSelectedItem

 @Html.EJS().MultiSelect("default").HideSelectedItem(true).Render() |

| Get Selected Item | **Method** : getSelectedItem()

@Html.EJ().DropDownList("selectCompany")

\$('#dropdown').ejDropDownList('getSelectedItem') | **Property** : value

@Html.EJS().MultiSelect("default").Render()

var mulObj =
 document.getElementById('multiselect').ej2_Instances[0];

 mulObj.value |

| Get Selected Value | **Method** : getSelectedValue()

@Html.EJ().DropDownList("selectCompany")

\$('#dropdown').ejDropDownList('getSelectedValue') | **Property** : value

@Html.EJS().MultiSelect("default").Render()

var mulObj =
 document.getElementById('multiselect').ej2_Instances[0];

 mulObj.value |

| Select Items By Indices | **Method** : selectItemsByIndices()

@Html.EJ().DropDownList("selectCompany")

var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.selectItemsByIndices("2,3") | Not Applicable |

| Select Item By Text | **Method** : selectItemByText()

@Html.EJ().DropDownList("selectCompany")

var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.selectItemByText("Computer IT ,Comics ") | Not Applicable |

| Select Item By Value | **Method** : selectItemByValue()

@Html.EJ().DropDownList("selectCompany")

var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.selectItemByValue("Computer IT ,Comics ") | Not Applicable |

| Un select Items By Indices | **Method** : unselectItemsByIndices()

@Html.EJ().DropDownList("selectCompany")

var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.unselectItemsByIndices ("2,3") | Not Applicable |

| Un select Item By Text | **Method** : unselectItemByText()

@Html.EJ().DropDownList("selectCompany")

var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.unselectItemByText ("Computer IT ,Comics ") | Not Applicable |

| Un select Item By Value | **Method** : unselectItemByValue()

@Html.EJ().DropDownList("selectCompany")

var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.unselectItemByValue ("Computer IT ,Comics ") | Not Applicable |

| Selected All | **Method** : checkAll()

@Html.EJ().DropDownList("selectCompany")

 var dropdown = \$("#dropdown ").data("ejDropDownList"); dropdown.checkAll() | **Method** :
 selectAll

@Html.EJS().MultiSelect("default").Render()

var mulObj =
 document.getElementById('multiselect').ej2_Instances[0];

 mulObj.selectAll(true) |

| Un Selected All | **Method** : unCheckAll()

@Html.EJ().DropDownList("selectCompany")

 var dropdown = \$("#dropdown
 ").data("ejDropDownList"); dropdown.unCheckAll() | **Method** : selectAll

@Html.EJS().MultiSelect("default").Render()

var mulObj =
 document.getElementById('multiselect').ej2_Instances[0];

 mulObj.selectAll(false) |

Common

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Virtual Scrolling | **Property** : AllowVirtualScrolling

@Html.EJ().DropDownList("default").AllowVirtualScrolling(true) | It will be achieved in
 sample level using actionComplete event |

| Virtual Scroll Mode | **Property** : VirtualScrollMode

@Html.EJ().DropDownList("default").AllowVirtualScrolling(true).VirtualScrollMode(Virtual
 ScrollMode.Normal) | Not Applicable |

| Custom class | **Property** : CssClass

@Html.EJ().DropDownList("default").CssClass('customstyle') | **Property** : CssClass

@Html.EJS().MultiSelect("default").CssClass('customstyle').Render() |

| Delimiter Char | **Property** : DelimiterChar

@Html.EJ().DropDownList("default").DelimiterChar('.') | **Property** : DelimiterChar

@Html.EJS().MultiSelect("default").DelimiterChar('.').Render() |

| Enable | **Property** : Enable @Html.EJ().DropDownList("default").Enable(true) |
Property : Enabled

@Html.EJS().MultiSelect("default").Enabled(true).Render() |

| Persistence | **Property** : EnablePersistence

@Html.EJ().DropDownList("default").EnablePersistence(true) | **Property** : EnablePersistence

@Html.EJS().MultiSelect("default").EnablePersistence(true).Render() |

| Load On Demand | **Property** : LoadOnDemand

@Html.EJ().DropDownList("default").LoadOnDemand(true) | By default, provided load on
 demand support |

| Height | **Property** : Height

@Html.EJ().DropDownList("default").Height(100) | Not
 Applicable |

| Html Attributes | **Property** : HtmlAttributes

@Html.EJ().DropDownList("default").HtmlAttributes((IDictionary<string,object>)ViewData[
 "HtmlAttrData"]) | **Property** : HtmlAttributes

@Html.EJS().MultiSelect("default").HtmlAttributes((IDictionary<string,object>)ViewData.h
 tmlAttrData).Render() |

| Width | **Property** : Width

@Html.EJ().DropDownList("default").Width(500) | **Property**
 : Width

@Html.EJS().MultiSelect("default").Width(400).Render() |

| Mode | **Property** : MultiSelectMode

@Html.EJ().DropDownList("default").MultiSelectMode(MultiSelectModeTypes.Delimiter) |
Property : Mode

 1. Delimeter

 2. Box

 3. Default

 4.
 CheckBox

 @Html.EJS().MultiSelect("default").Mode('Delimeter').Render() |

| Read Only | **Property** : ReadOnly

@Html.EJ().DropDownList("default").ReadOnly(true)
 | **Property** : Readonly @Html.EJS().MultiSelect("default").Readonly(true).Render() |

| Checkbox | **Property** : ShowCheckbox

@Html.EJ().DropDownList("default").ShowCheckbox(true) | **Property** : Mode

 @Html.EJS().MultiSelect("default").Mode('CheckBox').Render() |

| Rounded Corner | **Property** : ShowRoundedCorner

@Html.EJ().DropDownList("default").ShowRoundedCorner(true) | Not Applicable |

| Target ID | **Property** : TargetID

@Html.EJ().DropDownList("DropDownList1").TargetID("mail")<div id="mail"><div
 class="mailtools categorize"></div>Categorize and Move</div> | Not Applicable |

| Text | **Property** : Text @Html.EJ().DropDownList("DropDownList1").Text("Employee Name") | **Property** : Text @Html.EJS().MultiSelect("default").Text("Employee Name").Render() |

| Validation Message | **Property** : ValidationMessage @Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewData["DropDownSource"]).DropDownListFields(Df => Df.Text("Text").Value("Value")).ValidationMessage(message => message.AddMessage("required", "* Required").AddMessage("min", "Select > 30")) | The default error message for a rule can be customizable by defining it along with concern rule. |

| Validation Rules | **Property** : ValidationRules @Html.EJ().DropDownList("DropDownList1").Datasource((IEnumerable<Data>)ViewData["DropDownSource"]).DropDownListFields(Df => Df.Text("Text").Value("Value")).ValidationRules(vr => vr.AddRule("required", true).AddRule("min", 30)) | Use Form validator to validate the multiselect component and set validation rules. |

| Value | **Property** : Value @Html.EJ().DropDownList("DropDownList1").Value("Employee Value") | **Property** : Value @Html.EJS().MultiSelect("default").Value(ViewBag.value).Render() |

| Watermark Text | **Property** : WatermarkText @Html.EJ().DropDownList("DropDownList1").WatermarkText("Select employee") | **Property** : Placeholder @Html.EJS().MultiSelect("default").Placeholder('Select employee').Render() |

| Custom Value | Not Applicable | **Property** : AllowCustomValue @Html.EJS().MultiSelect("default").AllowCustomValue(true).Render() |

| Clear Button | Not Applicable | **Property** : ShowClearButton @Html.EJS().MultiSelect("default").ShowClearButton(true).Render() |

| DropDown Icon | Not Applicable | **Property** : ShowDropDownIcon @Html.EJS().MultiSelect("default").ShowDropDownIcon(true).Render() |

| Show Select All | Not Applicable | **Property** : ShowSelectAll @Html.EJS().MultiSelect("default").ShowSelectAll(true).Render() |

| z-Index | Not Applicable | **Property** : ZIndex @Html.EJS().MultiSelect("default").ZIndex(1000).Render() |

| Add Item | **Method** : addItem(object) @Html.EJ().DropDownList("DropDownList1") var dropdown = \$("#dropdown").data("ejDropDownList"); dropdown.addItem({ text : "new item"}); | **Method** : addItem(object) @Html.EJS().MultiSelect("default").Render() var mulObj = document.getElementById('multiselect').ej2_Instances[0]; mulObj.addItem({ text : "new item"}) |

| Clear Text | **Method** : clearText()

 @Html.EJ().DropDownList("DropDownList1")

 var dropdown = \$("#dropdown").data("ejDropDownList"); dropdown.clearText(); | **Property** :

```

value <br/> <br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.value = null |

| Destroy | Method : destroy() <br/> <br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/>
var dropdown = $("#dropdown ").data("ejDropDownList"); dropdown.destroy(); | Property : destroy
<br/> <br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.destroy() |

| Disable | Method : disable() <br/> <br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/>
var dropdown = $("#dropdown ").data("ejDropDownList"); dropdown.disable(); | Property : Enabled
<br/> <br/>@Html.EJS().MultiSelect("default").Enabled(false).Render() /> |

| Enable | Method : enable() <br/> <br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/>
var dropdown = $("#dropdown ").data("ejDropDownList"); dropdown.enable(); | Property : Enabled
<br/> <br/>@Html.EJS().MultiSelect("default").Enabled(true).Render() |

| Disable Items By Indices | Method : disableItemsByIndices() <br/>
<br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/> var dropdown = $("#dropdown
").data("ejDropDownList"); dropdown.disableItemsByIndices("3,4,5"); | Not Applicable |

| Enable Items By Indices | Method : enableItemsByIndices() <br/>
<br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/> var dropdown = $("#dropdown
").data("ejDropDownList"); dropdown.enableItemsByIndices("3,4,5"); | Not Applicable |

| Get Item Data By Value | Method : getItemDataByValue("value") <br/>
<br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/> var dropdown = $("#dropdown
").data("ejDropDownList"); dropdown.getItemDataByValue("value"); | Method :
getDataByValue("value") <br/> <br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var
mulObj = document.getElementById('multiselect').ej2_Instances[0];<br/><br/>
mulObj.getDataByValue("value") |

| Get List Data | Method : getListData() <br/>
<br/>@Html.EJ().DropDownList("DropDownList1")<br/> <br/> var dropdown = $("#dropdown
").data("ejDropDownList"); dropdown.getListData(); | Not Applicable |

| Hide Spinner | Not Applicable | Method : hideSpinner() <br/>
<br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.hideSpinner() |

| Show Spinner | Not Applicable | Method : showSpinner() <br/>
<br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.showSpinner() |

| Refresh | Not Applicable | Method : refresh() <br/>
<br/>@Html.EJS().MultiSelect("default").Render()<br/> <br/>var mulObj =
document.getElementById('multiselect').ej2_Instances[0];<br/><br/> mulObj.refresh() |

| Change | Event : Change<br/>
<br/>@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e => e.Change("onChange"))
| Event : Change<br/> <br/>@Html.EJS().MultiSelect("list").Change("onChange").Render() |

```

| Check Change | **Event** : CheckChange

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.CheckChange("onCheckChange")) | Not Applicable |

| Create | **Event** : Create

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e => e.Create("onCreate")) |
Event : Created

@Html.EJS().MultiSelect("list").Created("onCreated").Render() |

| Destroy | **Event** : Destroy

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.Destroy("onDestroy")) | **Event** : Destroyed

@Html.EJS().MultiSelect("list").Destroyed("onDestroyed").Render() |

| Focus In | **Event** : FocusIn

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.FocusIn("onFocusIn")) | **Event** : Focus

@Html.EJS().MultiSelect("list").Focus("onFocus").Render() |

| Focus Out | **Event** : FocusOut

@Html.EJ().DropDownList("selectCompany").ClientSideEvents(e =>
 e.FocusOut("onFocusOut")) | **Event** : Blur

@Html.EJS().MultiSelect("list").Blur("onBlur").Render() |

| Chip Selection | Not Applicable | **Event** : ChipSelection

@Html.EJS().MultiSelect("list").ChipSelection("onChipSelection").Render() |

| Custom Value Selection | Not Applicable | **Event** : CustomValueSelection

@Html.EJS().MultiSelect("list").CustomValueSelection("onCustomValueSelection").Render
 () |

| Removed | Not Applicable | **Event** : Removed

@Html.EJS().MultiSelect("list").Removed("onRemoved").Render() |

| Removing | Not Applicable | **Event** : Removing

@Html.EJS().MultiSelect("list").Removing("onRemoving").Render() |

| Tagging | Not Applicable | **Event** : Tagging

@Html.EJS().MultiSelect("list").Tagging("onTagging").Render() |

Numeric TextBox

Getting Started with ASP.NET MVC NumericTextBox Control

This section briefly explains about how to include [ASP.NET MVC NumericTextBox](#) control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

Note: Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

Note: If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/_Layout.cshtml** file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC NumericTextBox control

Now, add the Syncfusion ASP.NET MVC NumericTextBox control in `~/Views/Home/Index.cshtml` page.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Value(10).Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC NumericTextBox control will be rendered in the default web browser.



Range validation

You can set the minimum and maximum range of values in the NumericTextBox using the [Min](#) and [Max](#) properties, so the numeric value should be in the min and max range.

The validation behavior depends on the [StrictMode](#) property.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Value(16).Min(10).Max(20).Step(2).Render()
```

Formatting the value

User can set the format of the NumericTextBox control using [Format](#) property. The value will be displayed in the specified format, when the control is in focused out state. For more information about formatting the value, refer to this [link](#).

The below example demonstrates format the value by using currency format value `c2`.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Format("c2").Value(10).Render()
```



Precision of numbers

You can restrict the number of decimals to be entered in the NumericTextBox by using the [Decimals](#) and [ValidateDecimalOnType](#) properties. So, you can't enter the number whose precision is greater than the mentioned decimals.

- If `validateDecimalOnType` is false, number of decimals will not be restricted. Else, number of decimals will be restricted while typing in the NumericTextBox.

CSHTML

```
<div id='container'>
  <div class='wrap'>

@Html.EJS().NumericTextBox("strict").Format("n3").Value(10).ValidateDecimalOnType(true).Decimals(3).Placeholder("ValidateDecimalOnType Enabled").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
  </div>
  <div class='wrap'>

@Html.EJS().NumericTextBox("allow").Format("n3").Value(10).Decimals(3).Placeholder("ValidateDecimalOnType Disabled").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
  </div>
</div>
```

ValidateDecimalOnType Enabled
10.000

ValidateDecimalOnType Disabled
10.000

Note: [View Sample in GitHub.](#)

See also

- [How to perform custom validation using FormValidator](#)
- [How to customize the UI appearance of the control](#)
- [How to customize the spin button's up and down arrow](#)
- [How to customize the step value and hide spin buttons](#)
- [How to prevent nullable input in NumericTextBox](#)
- [How to maintain trailing zeros in NumericTextBox](#)

Number Formats in NumericTextBox Control

You can format the value of NumericTextBox using `format` property. The value will be displayed in the specified format when the control is in focused out state. The format string supports both the [standard numeric format string](#) and [custom numeric format string](#).

Standard formats

From the [standard numeric format](#), you can use the numeric related format specifiers such as `n`, `p` and `c` in the NumericTextBox control. By using these format specifiers, you can achieve the percentage and currency textbox behavior also.

The below example demonstrates percentage and currency formats.

CSHTML

```
<div id='container'>
    <div class='wrap'>

@Html.EJS().NumericTextBox("percent").Format("p2").Value(0.5).Min(0).Max(1).
Step(0.01).Placeholder("Percentage
format").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
    </div>
    <div class='wrap'>

@Html.EJS().NumericTextBox("currency").Format("c2").Value(10).Placeholder("C
urrency
format").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
    </div>
</div>
```

STANDARD.CS

```
public ActionResult Standard()
{
    return View();
}
```

Output be like the below.

**Custom formats**

From the [custom numeric format string](#), you can provide any custom format by combining one or more custom specifiers.

The below examples demonstrate format the value by using currency format string **#** and **0**.

CSHTML

```
<div id='container'>
    <div class='wrap'>

@Html.EJS().NumericTextBox("numeric").Format("###.##").Value(10).Placeholder
("Custom format string
#").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
    </div>
    <div class='wrap'>

@Html.EJS().NumericTextBox("numeric1").Format("000.00").Value(10).Placeholde
```

```

r("Custom format string
0").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
</div>
</div>

```

CUSTOM.CS

```

public ActionResult Custom()
{
    return View();
}

```

Output be like the below.



Globalization

Internationalization

Internationalization library provides support for formatting and parsing the number by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data. The `NumericTextBox` comes with built-in internationalization support to adapt based on culture. For more information about internationalization, refer to this [link](#).

By default, all the Essential JS 2 control are specific to English culture (en-US).

If you want to go with the different culture other than `English`, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this [link](#).

,

```
npm install cldr-data --save
```

,

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

In ASP.NET MVC refer the culture files directly from `/node_modules/cldr-data` location as like the below code examples

```

`typescript
function loadCultureFiles(name) {
var files = ['currencies.json', 'numbers.json'];

```

```

if (name === 'zh') {
files.push('currencyData.json');
}
var loader = ej.base.loadCldr;
var loadCulture = function () {
var val, ajax;
if (name === 'zh' && prop === files.length - 1) {
ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-
data/supplemental/' + files[prop], 'GET', false);
} else {
ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-data/main/' +
name + '/' + files[prop], 'GET', false);
}
ajax.onSuccess = function (value) {
val = value;
};
ajax.send();
loader (JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
loadCulture();
}
}
loadCultureFiles('de');
`

```

Localization

[Localization](#) library allows users to localize the default text contents of the NumericTextBox to different cultures using the [locale](#) property.

In NumericTextBox, spin buttons title for the tooltip will be localized based on the culture.

| | | | | |
|--|----------------|--|-----------------|--|
| | Locale key | | en-US (default) | |
| | ----- | | ----- | |
| | incrementTitle | | Increment value | |
| | decrementTitle | | Decrement value | |

- Before changing to a culture other than English, ensure that locale text for the concerned culture is loaded through `load` method of `L10n` class.

```
`typescript
ej.base.L10n.load({
  'en': {
    'numerictextbox': { incrementTitle: 'Increment value', decrementTitle: 'Decrement value' }
  },
  'de': {
    'numerictextbox': { incrementTitle: 'Wert erhöhen', decrementTitle: 'Dekrementwert' }
  },
  'zh': {
    'numerictextbox': { incrementTitle: '增值', decrementTitle: '遞減值' }
  }
});
`
```

CSHTML

```
<div class="content-wrapper">
  <div class="divelement">

    @Html.EJS().NumericTextBox("numeric").Value(10).Locale("de").Placeholder("Ge
    ben Sie den Wert
    ein").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
  </div>
  <div class="divelement">

    @Html.EJS().NumericTextBox("percentage").Format("p2").Value(0.5).Locale("de"
    ).Max(1).Min(0).Step(0.01).Placeholder("Geben Sie den Prozentsatz
    ein").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
  </div>
  <div class="divelement">

    @Html.EJS().NumericTextBox("currency").Format("c2").Value(100).Locale("de")
    .Currency("EUR").Placeholder("Geben Sie die Währung
    ein").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
  </div>
  <div class="wrapper">

    @Html.EJS().DropDownList("cultures").Text("de").Change("changeLocale").Place
    holder("Choose
    Culture").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).DataSo
    urce(ViewBag.cultureData).Fields(new
    Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CultureName"
    }).Render()
  </div>
</div>
```

```

</div>
<script>
    function loadCultureFiles(name) {
        var files = ['currencies.json', 'numbers.json'];
        if (name === 'zh') {
            files.push('currencyData.json');
        }
        var loader = ej.base.loadCldr;
        var loadCulture = function (prop) {
            var val, ajax;
            if (name === 'zh' && prop === files.length - 1) {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../.../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
            } else {
                ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../.../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
            }
            ajax.onSuccess = function (value) {
                val = value;
            };
            ajax.send();
            loader(JSON.parse(val));
        };
        for (var prop = 0; prop < files.length; prop++) {
            loadCulture(prop);
        }
    }
    loadCultureFiles('de');
    function changeLocale() {
        var numeric = document.getElementById("numeric").ej2_instances[0];
        var percent =
document.getElementById("percentage").ej2_instances[0];
        var currency = document.getElementById("currency").ej2_instances[0];
        var culture = document.getElementById('cultures').value;
        numeric.locale = culture;
        percent.locale = culture;
        currency.locale = culture;
        if (culture !== 'en') {
            loadCultureFiles(culture);
        }
        if (culture === 'zh') {
            currency.currency = 'CNY';
            numeric.placeholder = '输入值';
            currency.placeholder = '输入货币';
            percent.placeholder = '输入百分比';
        }
        else if (culture === 'de') {
            currency.currency = 'EUR';
            numeric.placeholder = 'Geben Sie den Wert ein';
            currency.placeholder = 'Geben Sie die Währung ein';
            percent.placeholder = 'Geben Sie den Prozentsatz ein';
        }
        else {
            currency.currency = 'USD';
        }
    }

```

```

        numeric.placeholder = 'Enter the value';
        currency.placeholder = 'Enter the currency';
        percent.placeholder = 'Enter the percentage';
    }
}
</script>
<style>
    .content-wrapper {
        width: 25%;
        margin: 0 auto;
    }
    .divelement {
        padding: 10px;
    }
    .wrapper {
        width: 50%;
        padding-top: 50px;
        margin: 0 auto;
    }
</style>
<script>
    ej.base.enableRipple(true)
    ej.base.L10n.load({
        'en': {
            'numerictextbox': { incrementTitle: 'Increment value',
decrementTitle: 'Decrement value' }
        },
        'de': {
            'numerictextbox': { incrementTitle: 'Wert erhöhen',
decrementTitle: 'Dekrementwert' }
        },
        'zh': {
            'numerictextbox': { incrementTitle: '增值', decrementTitle: '遞減
值' }
        },
    });
</script>

```

LOCALIZATION.CS

```

public class HomeController : Controller
{
    public class Cultures
    {
        public string CultureName { get; set; }
    }
    public IActionResult Index()
    {
        List<Cultures> cultureData = new List<Cultures>();
        cultureData.Add(new Cultures() { CultureName = "de" });
        cultureData.Add(new Cultures() { CultureName = "zh" });
        cultureData.Add(new Cultures() { CultureName = "en" });
        ViewBag.cultureData = cultureData;
        return View();
    }
}

```

Output be like the below.

The image shows three instances of the NumericTextBox control, each with a label in Chinese and a value displayed in the text box. The first control is labeled '输入值' (Input Value) and shows '10.00'. The second is labeled '输入百分比' (Input Percentage) and shows '50.00%'. The third is labeled '输入货币' (Input Currency) and shows '¥ 100.00'. Each control has a horizontal line below the text box and two small triangular arrows (down and up) to its right. Below these controls is a 'Choose Culture' dropdown menu with 'zh' selected.

Right to Left

RTL provides an option to switch the text direction and layout of the NumericTextBox control from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL NumericTextBox, set the [enableRtl](#) to true.

CSHTML

```
<div class="content-wrapper">
  <div class="divelement">

    @Html.EJS().NumericTextBox("numeric").Value(10).Locale("de").Placeholder("Geben Sie den Wert ein").EnableRtl(true).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
  </div>
  <div class="divelement">

    @Html.EJS().NumericTextBox("percentage").Format("p2").Value(0.5).Locale("de").Max(1).Min(0).Step(0.01).Placeholder("Geben Sie den Prozentsatz ein").EnableRtl(true).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
  </div>
  <div class="divelement">

    @Html.EJS().NumericTextBox("currency").Format("c2").Value(100).Locale("de").Currency("EUR").Placeholder("Geben Sie die Währung ein").EnableRtl(true).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
  </div>
</div>
```



```

<div class="wrapper">

@Html.EJS().DropDownList("cultures").Text("de").Change("changeLocale").Place
holder("Choose
Culture").EnableRtl(true).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelTyp
e.Always).DataSource(ViewBag.cultureData).Fields( new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "CultureName"
}).Render()

</div>
</div>
<script>
function loadCultureFiles(name) {
    var files = ['currencies.json', 'numbers.json'];
    if (name === 'zh') {
        files.push('currencyData.json');
    }
    var loader = ej.base.loadCldr;
    var loadCulture = function (prop) {
        var val, ajax;
        if (name === 'zh' && prop === files.length - 1) {
            ajax = new ej.base.Ajax(location.origin +
location.pathname + '/../../node_modules/cldr-data/supplemental/' +
files[prop], 'GET', false);
        } else {
            ajax = new ej.base.Ajax(location.origin +
location.pathname + '/../../node_modules/cldr-data/main/' + name + '/' +
files[prop], 'GET', false);
        }
        ajax.onSuccess = function (value) {
            val = value;
        };
        ajax.send();
        loader(JSON.parse(val));
    };
    for (var prop = 0; prop < files.length; prop++) {
        loadCulture(prop);
    }
}
loadCultureFiles('de');
function changeLocale() {
    var numeric =
document.getElementById("numeric").ej2_instances[0];
    var percent =
document.getElementById("percentage").ej2_instances[0];
    var currency =
document.getElementById("currency").ej2_instances[0];
    var culture = document.getElementById('cultures').value;
    numeric.locale = culture;
    percent.locale = culture;
    currency.locale = culture;
    if (culture !== 'en') {
        loadCultureFiles(culture);
    }
    if (culture === 'zh') {
        currency.currency = 'CNY';
        numeric.placeholder = '输入值';
    }
}

```

```

        currency.placeholder = '输入货币';
        percent.placeholder = '输入百分比';
    }
    else if (culture === 'de') {
        currency.currency = 'EUR';
        numeric.placeholder = 'Geben Sie den Wert ein';
        currency.placeholder = 'Geben Sie die Währung ein';
        percent.placeholder = 'Geben Sie den Prozentsatz ein';
    }
    else {
        currency.currency = 'USD';
        numeric.placeholder = 'Enter the value';
        currency.placeholder = 'Enter the currency';
        percent.placeholder = 'Enter the percentage';
    }
}
</script>
<style>
    .content-wrapper {
        width: 25%;
        margin: 0 auto;
    }
    .divelement {
        padding: 10px;
    }
    .wrapper {
        width: 50%;
        padding-top: 50px;
        margin: 0 auto;
    }
</style>
<script>
    ej.base.enableRipple(true)
    ej.base.L10n.load({
        'en': {
            'numerictextbox': { incrementTitle: 'Increment value',
decrementTitle: 'Decrement value' },
        },
        'de': {
            'numerictextbox': { incrementTitle: 'Wert erhöhen',
decrementTitle: 'Dekrementwert' },
        },
        'zh': {
            'numerictextbox': { incrementTitle: '增值', decrementTitle:
'遞減值' },
        },
    });
</script>

```

RTL.CS

```

public class HomeController : Controller
{
    public class Cultures
    {

```

```

        public string CultureName { get; set; }
    }
    public ActionResult Index()
    {
        List<Cultures> cultureData = new List<Cultures>();
        cultureData.Add(new Cultures() { CultureName = "de" });
        cultureData.Add(new Cultures() { CultureName = "zh" });
        cultureData.Add(new Cultures() { CultureName = "en" });
        ViewBag.cultureData = cultureData;
        return View ();
    }
    ...
}

```

Output be like the below.

The screenshot displays a web form with four input elements. The first three are numeric textboxes, each with a label above it and a value to its right. The first is labeled 'Geben Sie den Wert ein' with a value of '10,00'. The second is labeled 'Geben Sie den Prozentsatz ein' with a value of '% 50,00'. The third is labeled 'Geben Sie die Währung ein' with a value of '€ 100,00'. Each of these three textboxes has small upward and downward arrow icons to its left. The fourth element is a dropdown menu labeled 'Choose Culture' with the value 'de' selected and displayed to its right.

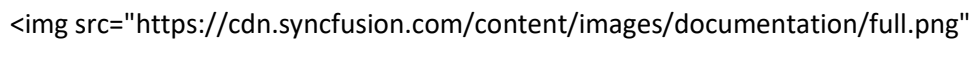
Accessibility

The Numerictextbox component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Numerictextbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) |  alt="Yes"> |

| [Section 508 Support](#) |  alt="Yes"> |

```

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| Accessibility Checker Validation |  |

| Axe-core Accessibility Validation |  |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

WAI-ARIA attributes

The NumericTextBox characterized with complete ARIA Accessibility support which helps to accessible by on-screen readers and other assistive technology devices. This component designed with the reference of the guidelines document given in [WAI ARAI Accessibility practices](#).

The NumericTextBox uses the `spinbutton` role and following ARIA properties to its element based on its state.

| Property | Functionality |

| --- | --- |

| aria-live | The `aria-live` attribute indicates the priority of updates to a live region |

| aria-valuemin | The `aria-valuemin` property specifies the minimum allowable range of the NumericTextBox. |

| **aria-valuemax** | The **aria-valuemax** property specifies the maximum allowable range of the NumericTextBox. |

| **aria-disabled** | The **aria-disabled** property indicates disabled state of the NumericTextBox. |

| **aria-readonly** | The **aria-readonly** property indicates the read-only state of the NumericTextBox. |

| **aria-valuenow** | The **aria-valuenow** property specifies the current value of the NumericTextBox. |

| **aria-invalid** | The **aria-invalid** property indicates that the user input is incorrect or not within acceptable ranges. |

| **aria-label** | The **aria-label** property indicates a string value that labels the NumericTextBox. |

Keyboard interaction

Keyboard interaction of the NumericTextBox control has been designed based on [WAI-ARIA Practices](#) described for the NumericTextBox and it is an alternative to mouse actions to interact with the NumericTextBox.

The below table shows shortcut keys and its corresponding usage.

| Keyboard shortcuts | Actions |
|--------------------|-----------------------|
| --- --- | |
| Arrow Down | Increments the value. |
| Arrow Up | Decrements the value |

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Value(10).Render()
```

ACCESSIBILITY.CS

```
public ActionResult Accessibility()
{
    return View();
}
```

Output be like the below.



Ensuring accessibility

The NumericTextBox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the NumericTextBox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the NumericTextBox component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

Style and appearance in NumericTextBox Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

Customizing the appearance of NumericTextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
/ To specify height and font size /

.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-input, .e-input-group.e-control-wrapper textarea.e-input {
height: 40px;
font-size: 20px;
}
```

Customizing the NumericTextBox icons

Use the following CSS to customize the Numeric TextBox icons

```
`css
/ To specify font size and background color /

.e-numeric.e-control-wrapper.e-input-group .e-input-group-icon {
font-size: 20px;
background-color: beige;
}
```

How To

Customize the UI appearance of the control

You can change the appearance of the NumericTextBox by adding custom `cssClass` to the control and enabling styles. Refer to the following example to change the appearance of the NumericTextBox.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Value(10).Placeholder("numeric").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).CssClass("e-style").Render()
<style>
    .e-numeric.e-style .e-control.e-numerictextbox {
        color: royalblue;
        font-size: xx-large;
        border: 0px;
    }
    .e-input-group.e-control-wrapper:not(.e-success):not(.e-warning):not(.e-error):not(.e-float-icon-left), .e-float-input.e-control-wrapper:hover:not(.e-success):not(.e-warning):not(.e-error):not(.e-disabled):not(.e-float-icon-left) {
        border-color: royalblue;
    }
```

```

    }
    .e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-down {
        color: royalblue;
    }
    .e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::before
    {
        background: royalblue;
    }
    .e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::after
    {
        background: royalblue;
    }
    .e-control-wrapper.e-numeric.e-float-input.e-style .e-spin-up {
        color: royalblue;
    }
    .e-control-wrapper.e-numeric.e-float-input.e-style.e-input-group .e-
float-text.e-label-top {
        color: royalblue;
        font-size: medium;
    }
</style>

```

CUSTOMCSS.CS

```

public ActionResult customCss()
{
    return View();
}

```

Output be like the below.



Customize the spin button's up and down arrow

This section explains about how to change/customize spin up and down icons. You can customize spin button icons using `e-spin-up` and `e-spin-down` classes of those buttons.

You can override the default icons of `e-spin-up` and `e-spin-down` classes using the following CSS code snippets.

```

`css
.e-numeric .e-input-group-icon.e-spin-up:before {
content: "\e823";
color: rgba(0, 0, 0, 0.54);
}
.e-numeric .e-input-group-icon.e-spin-down:before {

```

```
content: "\e934";
color: rgba(0, 0, 0, 0.54);
}
```

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Value(10).Render()
<style>
  /* csslint ignore:start */
  .e-numeric .e-input-group-icon.e-spin-up:before {
    content: "\e823";
    color: rgba(0, 0, 0, 0.54);
  }
  .e-numeric .e-input-group-icon.e-spin-down:before {
    content: "\e934";
    color: rgba(0, 0, 0, 0.54);
  }
  /* csslint ignore:end */
</style>
```

BUTTONS.CS

```
public ActionResult Buttons()
{
    return View();
}
```

Output be like the below.



Customize the step value and hide spin buttons

The spin buttons allow you to increase or decrease the value with the predefined [step](#) value. The visibility of spin buttons can be set using the [showSpinButton](#) property.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Value(16).Min(10).Max(100).ShowSpinButton(false).Render()
```

ICONS.CS

```
public ActionResult Icons()
{
    return View();
}
```

Output be like the below.

16.00

Maintain trailing zeros in NumericTextBox

By default, trailing zeros disappear when the NumericTextBox gets focus. However, you can use the following sample to maintain the trailing zeros while focusing the NumericTextBox.

CSHTML

```
@Html.EJS().NumericTextBox("numeric").Format("n2").Value(10).Decimals(2).Change("numericFocus").Created("onCreate").Render()
<script>
    function numericFocus() {
        var numericObj = this.ej2_instances ? this.ej2_instances[0] : this;
        numericObj.element.value =
        numericObj.formattedValue(numericObj.decimals, +numericObj.element.value);
    }
    function onCreate() {
        document.getElementById('numeric').addEventListener('focus',
        numericFocus);
    }
</script>
```

TRAILINGZEROS.CS

```
public ActionResult nullableInput()
{
    return View();
}
```

NumericTextBoxFor and Model Binding

This section demonstrates the Strongly typed extension support in NumericTextBox. The view which bind with any model is called as strongly typed view. You can bind any class as model to view. You can access model properties on that view. You can use data associated with model to render controls.

In this sample, first click the submit button to post the selected value in the MaskedTextBox. When posting the null value, validation error message will be shown below the NumericTextBox.

CSHTML

```
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)
    <div class="col-lg-12 control-section">
        <div id="wrapper">
            @Html.EJS().NumericTextBoxFor(model =>
            model.value).Width("200px").Render()
            <div>
                @Html.ValidationMessageFor(model => model.value)
            </div>
            <div id="submitButton">
                @Html.EJS().Button("btn").Content("Post").Render()
            </div>
        </div>
    </div>
}
```

```
}  
<style>  
    #wrapper {  
        max-width: 246px;  
        margin: 30px auto;  
        padding-top: 50px;  
    }  
    #submitbutton {  
        margin-top: 20px;  
        margin-left: 65px;  
    }  
    #control-content #wrapper .field-validation-error {  
        color: red;  
    }  
</style>
```

MODELBINDING.CS

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Threading.Tasks;  
using Microsoft.AspNetCore.Mvc;  
using System.ComponentModel.DataAnnotations;  
namespace EJ2CoreSampleBrowser.Controllers  
{  
    public class NumericValue  
    {  
        [Required]  
        public int value { get; set; }  
    }  
    public partial class NumericTextBoxController : Controller  
    {  
        public ActionResult DefaultFunctionalities()  
        {  
            NumericValue val = new NumericValue();  
            val.value = 10;  
            return View(val);  
        }  
        [HttpPost]  
        public ActionResult DefaultFunctionalities(NumericValue model)  
        {  
            NumericValue val = new NumericValue();  
            val.value = model.value;  
            return View(val);  
        }  
    }  
}
```

Output be like the below.



Perform custom validation using FormValidator

This section explains how to perform custom validation on the NumericTextBox using FormValidator. The NumericTextBox will be validated when the value changes or the user clicks the submit button. Validation can be performed by adding custom validation in the rules collection of the FormValidator.

CSHTML

```
<form id="form-element">

@Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Change("onChange").Placeholder("numeric").Created("onCreate").StrictMode(false).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Render()
    @Html.EJS().Button("submit_btn").Content("submit").Render()
</form>
<script>
    function onCreate() {
        document.getElementById(this.element.id).setAttribute("name",
"numericRange");
    }
    function onChange(args) {
        if (numeric.value != null) {
            formObject.validate("numericRange");
        }
    }
    var customFn = function (args) {
        if (numeric.value >= 10 && numeric.value <= 100) {
            return true;
        }
        else {
            return false;
        }
    };
    // sets required property in the FormValidator rules collection
    var options = {
        rules: {
            'numericRange': { required: [true, "Number is required"] },
        },
    };
    // defines FormValidator to validate the MaskedTextBox
    var formObject = new ej.inputs.FormValidator('#form-element', options);
    // places error label outside the MaskedTextBox using the
customPlacement event of FormValidator
    formObject.customPlacement = function (element, errorElement) {
        element.parentNode.parentNode.appendChild(errorElement);
    };
    formObject.addRules('numericRange', { range: [customFn, "Please enter a
number between 10 to 100"] });
    document.getElementById("submit_btn").addEventListener('click', function
() {
```

```

        formObject.validate("numericRange");
        var ele = document.getElementById('numeric');
        // checks for incomplete value and alerts the formt submit
        if (ele.value !== "" && ele.value >= 10 && ele.value <= 100) {
            alert("Submitted");
        }
    });
</script>
<style>
    .e-numeric.e-control-wrapper {
        margin-bottom: 20px;
    }
    label.e-error {
        margin-top: -50px;
    }
</style>

```

CUSTOMVALIDATION.CS

```

public ActionResult customValidation()
{
    return View();
}

```

Output be like the below.



Prevent nullable input in NumericTextBox

By default, the value of the NumericTextBox sets to null. In some applications, the value of the NumericTextBox should not be null at any instance. In such cases, following sample can be used to prevent nullable input in NumericTextBox.

CSHTML

```

@Html.EJS().NumericTextBox("numeric").Created("onCreate").Blur("onBlur").Render()
<script>
    function onCreate(args) {
        if (this.value == null) {
            this.value = 0;
        }
    }
    function onBlur(args) {
        if (args.value == null) {
            numeric.value = 0;
        }
    }
</script>

```

NULLABLEINPUT.CS

```
public ActionResult nullableInput()
{
    return View();
}
```

Output be like the below.

[Migration from Essential JS 1](#)

This article describes the API migration process of NumericTextBox component from Essential JS 1 to Essential JS 2.

[Common](#)

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers on creation | **Event** *Create* @Html.EJ().NumericTextbox("numeric").Value(120).ClientSideEvents(s => s.Create("onCreate")) Script function onCreate(){ } | **Event:** *Created* @Html.EJS().NumericTextBox("numeric").Value(120).Created("onCreate").Render() Script function onCreate(){ } |

| Adding custom classes | **Property** *CssClass* @Html.EJ().NumericTextbox("numeric").Value(100).CssClass("custom") | **Property:** *CssClass* @Html.EJS().NumericTextBox("numeric").Value(100).CssClass("custom").Render() |

| Triggers when editor is destroyed | **Event** *Destroy* @Html.EJ().NumericTextbox("numeric").Value(120).ClientSideEvents(s => s.Destroy("onDestroy")) Script function onDestroy(){ } | **Event:** *Destroyed* @Html.EJS().NumericTextBox("numeric").Value(120).Destroyed("onDestroy").Render() Script function onDestroy(){ } |

| Destroys textbox | **Method** *destroy* @Html.EJ().NumericTextbox("numeric").Value(100) Script var numericObj = \$("#numeric").data("ejNumericTextbox"); numericObj.destroy(); | **Method:** *destroy* @Html.EJS().NumericTextBox("numeric").Value(100).Render() Script var numeric = document.getElementById('numeric').ej2_instances[0] numeric.destroy(); |

| Control state | **Property** *Enabled* @Html.EJ().NumericTextbox("numeric").Value(100).Enabled(false) | **Property:** *Enabled* @Html.EJS().NumericTextBox("numeric").Value(100).Enabled(false).Render() |

| Persistence | **Property** *EnablePersistence*

@Html.EJ().NumericTextbox("numeric").Value(100).EnablePersistence(true) | **Property:**

EnablePersistence

 />@Html.EJS().NumericTextBox("numeric").Value(100).EnablePersistence(true).Render() |

| Right To Left | **Property** *EnableRTL*

 />@Html.EJ().NumericTextbox("numeric").Value(100).EnableRTL(true) | **Property:** *EnableRtl*

 />@Html.EJS().NumericTextBox("numeric").Value(100).EnableRtl(true).Render() |

| Triggers when editor is focused in | **Event** *FocusIn*

 />@Html.EJ().NumericTextbox("numeric").
Value(20).ClientSideEvents(s => s.FocusIn("focusIn"))

 />**Script**
function focusIn(){ } | **Event:**
Focus

@Html.EJS().NumericTextBox("numeric").Value(100).InputMode(InputMode.Text).Focus
 s("onFocus")
Script
function onFocus(){ } |

| Triggers when editor is focused out | **Event** *FocusOut*

 />@Html.EJ().NumericTextbox("numeric").Value(100).
ClientSideEvents(s =>
 s.FocusOut("focusOut"))
Script
function focusOut(){ } | **Event:**
Blur

@Html.EJS().NumericTextBox("numeric").Value(100).InputMode(InputMode.Text).Blur("onBlur")

Script
function onBlur(){ } |

| Sets Height | **Property** *Height*

 />@Html.EJ().NumericTextbox("numeric").Value(100).Height("40px") | **Can be achieved using**,

 />
@Html.EJS().NumericTextBox("numeric").Value(100).CssClass("custom").Render()
Css

 />.e-numerictextbox.custom{
height: 40px;
} |

| HTML Attributes | **Property** *HtmlAttributes*

 />@Html.EJ().NumericTextbox("numeric").Value(100).HtmlAttributes(htmlAttr)

Script

 />
@
Dictionary<string, object> htmlAttr = new Dictionary<string, object>();

 />htmlAttr.Add("disabled", "disabled");
} | **Can be achieved**
using
@Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Created("onCreate").Render()

 />**Script**
function onCreate()
 {
document.getElementById(this.element.id).setAttribute("name", "numericRange");
} |

| Name of editor | **Property** *name*

 />@Html.EJ().NumericTextbox("numeric").Value(100).Name("Textbox") | **Can be achieved using**

 @Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Created("onCreate").Render()
Script

 />function onCreate() {
document.getElementById(this.element.id).setAttribute("name",
 "numericRange");
} |

| Read only | **Property** *ReadOnly*

 />@Html.EJ().NumericTextbox("numeric").Value(80).ReadOnly(true) | **Property:** *ReadOnly*

 />@Html.EJS().NumericTextBox("numeric").Value(80).ReadOnly(true).Render() |

| Rounded corners | **Property** *ShowRoundedCorner*

 />@Html.EJ().NumericTextbox("numeric").Value(100).ShowRoundedCorner(true) | **Can be achieved**
using
@Html.EJS().NumericTextBox("numeric").Created("onCreate").FloatLabelType(Syncfusion.EJ2
 .Inputs.FloatLabelType.Always).CssClass("e-style").Render()
Css
.e-control-wrapper.e-
 numeric.e-input-group.e-style {
border: solid 2.5px;
border-radius: 1rem;
padding-left:
 12px;
}
.e-control-wrapper.e-numeric.e-input-group.e-style.e-input-focus {
border: solid
 grey 2.5px !important;
border-radius: 1rem;
}
.e-control-wrapper.e-numeric.e-float-
 input.e-style .e-float-line::before,.e-control-wrapper.e-numeric.e-float-input.e-style .e-float-line::after
 {
background: none;
}
.e-control-wrapper.e-numeric.e-input-group.e-style.e-input-
 focus{
border: solid grey !important;
} |

| Spin Button | **Property** *ShowSpinButton*

 />@Html.EJ().NumericTextbox("numeric").Value(20).ShowSpinButton(true) | **Property:** *ShowSpinButton*

 />@Html.EJS().NumericTextBox("numeric").Value(20).ShowSpinButton(false).Render() |

| Width | **Property** *Width*

 />@Html.EJ().NumericTextbox("numeric").Value(20).width("220px") | **Property:** *Width*

 />@Html.EJS().NumericTextBox("numeric").Value(20).Width("220px").Render() |

| Clear Button | Not Applicable | **Property:** *ShowClearButton*

 />@Html.EJS().NumericTextBox("numeric").Value(100).ShowClearButton(true).Render() |

Globalization

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Localization culture | **Property** *Locale*

 />@Html.EJ().NumericTextbox("numeric").Value(80).Locale("de-DE") | **Property:** *Locale*

 />@Html.EJS().NumericTextBox("numeric").Value(80).Locale("de-DE").Render() |

Group

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Group digits in editor | **Property** *GroupSize*

 />@Html.EJ().NumericTextbox("numeric").Value(100).GroupSize("2") | Not Applicable |

| Group Separator | **Property** *GroupSeparator*

 />@Html.EJ().NumericTextbox("numeric").Value(100).GroupSeparator("-") | Not Applicable |

Numeric configuration

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Triggers on value change | **Event** *Change*

 />@Html.EJ().NumericTextbox("numeric").Value(120).
ClientSideEvents(s =>
 s.Change("onChange"))
Script
function onChange(){ } | **Event:** *Change*

 />@Html.EJS().NumericTextBox("numeric").Value(120).Change("onChange").Render()
Script

 />function onChange(){ } |

| Sets digits allowed after decimal point | **Property** *DecimalPlaces*

 />@Html.EJ().NumericTextbox("numeric").Value(100).DecimalPlaces(2) | **Property:** *Decimals*

 />@Html.EJS().NumericTextBox("numeric").Value(100).Format("n2").Decimals("2").Render() |

| Decrement value | Not Applicable | **Method:** *decrement*

 />@Html.EJS().NumericTextBox("numeric").Value(100).Render()

Script

var
 numeric = document.getElementById('numeric').ej2_instances[0]
numeric.decrement(); |

| Disable the textbox | **Method** *disable*

 />@Html.EJ().NumericTextbox("numeric").Value(200)
Script

var numericObj =
 \$("#numeric").data("ejNumericTextbox");
numericObj.disable(); | **Can be achieved**
using
@Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Created("onCreate").Render()

 />**Script**
function onCreate() {
var numeric = this;
numeric.enabled = false;
}
 |

| Enable the textbox | **Method** *enable*
 />@Html.EJ().NumericTextbox("numeric").Value(200)
<Script>

var numericObj =
 \$("#numeric").data("ejNumericTextbox");
numericObj.enable(); | **Can be achieved**
 using
@Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Created("onCreate").Render()
<Script>
function onCreate() {
var numeric = this;
numeric.enabled = true;
}
 |

| Gets value of editor | **Method** *getValue*
 />@Html.EJ().NumericTextbox("numeric").Value(100)
<Script>

var numericObj =
 \$("#numeric").data("ejNumericTextbox");
numericObj.getValue(); | **Method:** *getText*
 />@Html.EJS().NumericTextBox("numeric").Value(100).Render()

<Script>

var
 numeric = document.getElementById('numeric').ej2_instances[0]
numeric.getText(); |

| Increment value | Not Applicable | **Method:** *increment*
 />@Html.EJS().NumericTextBox("numeric").Value(80).Render()

<Script>

var
 numeric = document.getElementById('numeric').ej2_instances[0]
numeric.increment(); |

| Step value | **Property** *IncrementStep*
 />@Html.EJ().NumericTextbox("numeric").Value(100).IncrementStep(2) | **Property:** *Step*
 />@Html.EJS().NumericTextBox("numeric").Value(100).Step(2).Render() |

| Sets Maximum value | **Property** *MaxValue*
 />@Html.EJ().NumericTextbox("numeric").Value(100).MaxValue(200) | **Property:** *Max*
 />@Html.EJS().NumericTextBox("numeric").Value(100).Max(200).Render() |

| Sets Minimum value | **Property** *MinValue*
 />@Html.EJ().NumericTextbox("numeric").Value(100).MinValue(20) | **Property:** *Min*
 />@Html.EJS().NumericTextBox("numeric").Value(100).Min(20).Render() |

| Negative pattern for formatting values | **Property** *NegativePattern*
 />@Html.EJ().NumericTextbox("numeric").Value(-20).NegativePattern("(n)") | Not Applicable |

| Positive pattern for formatting values | **Property** *PositivePattern*
 />@Html.EJ().NumericTextbox("numeric").Value(20).PositivePattern("n kg") | Not Applicable |

| Specifies value | **Property** *Value*
 />@Html.EJ().NumericTextbox("numeric").Value(100) | **Property:** *value*
 />@Html.EJS().NumericTextBox("numeric").Value(100).Render() |

| Displays hint on editor | **Property** *WatermarkText*
 />@Html.EJ().NumericTextbox("numeric").Value(80).WatermarkText("Enter value") | **Property:**
Placeholder
 />@Html.EJS().NumericTextBox("numeric").Value(100).Placeholder("Enter
 value").Render() |

| Placeholder float type | Not Applicable | **Property:** *FloatLabelType*
 />@Html.EJS().NumericTextBox("numeric").Value(200).Placeholder("Enter
 value").FloatLabelType("Auto").Render() |

Number Formats

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Set Currency symbol | **Property** *CurrencySymbol*
 />@Html.EJ().CurrencyTextbox("currency").Value(100).CurrencySymbol("EUR") | **Property:** *Currency*
 />@Html.EJS().NumericTextBox("numeric").Format("c2").Value(100).Currency("EUR").Render() |

| Number Format | Not Applicable | **Property:** *Format* @Html.EJS().NumericTextBox("numeric").Format("n2").Value(200).Render() |

Validation

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Strict Mode | **Property** *EnableStrictMode* @Html.EJ().NumericTextbox("numeric").Value(80).EnableStrictMode(true) | **Property:** *StrictMode* @Html.EJS().NumericTextBox("numeric").StrictMode(true).Value(80).Render() |

| Validation on typing | **Property** *ValidateOnType* @Html.EJ().NumericTextbox("numeric").Value(100).ValidateOnType(true) | **Property:** *ValidateDecimalOnType* @Html.EJS().NumericTextBox("numeric").ValidateDecimalOnType(true).Value(100).Render() |

| Validation Message | **Property** *ValidationMessage* @Html.EJ().NumericTextbox("numeric").Value(100).ValidationRules(rule => rule.AddRule("required", true)).ValidationMessage(msg => msg.AddMessage("required", "Required value")) | **Can be achieved using** <form id="form-element"> @Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Placeholder("Enter the number").Change("onChange").Created("onCreate").StrictMode(false).Render() </form> function onCreate() { document.getElementById(this.element.id).setAttribute("name", "numericRange"); </br></br>function onChange() { if (numeric.value != null) { formObject.validate("numericRange"); } var options = { rules: { 'numericRange': { required: [true, "Number is required"] }, }, }; var formObject = new ej.inputs.FormValidator('#form-element', options); formObject.customPlacement = function (element, errorElement) { element.parentNode.parentNode.appendChild(errorElement); }; |

| Validation Rules | **Property** *ValidationRules* @Html.EJ().NumericTextbox("numeric").Value(100).ValidationRules(rule => rule.AddRule("required", true)) | **Can be achieved using** <form id="form-element"> @Html.EJS().NumericTextBox("numeric").Min(10).Max(100).Placeholder("Enter the number").Change("onChange").Created("onCreate").StrictMode(false).Render() </form> function onCreate() { document.getElementById(this.element.id).setAttribute("name", "numericRange"); </br></br>function onChange(args) { if (numeric.value != null) { formObject.validate("numericRange"); } var options = { rules: { 'numericRange': { required: [true] }, }, }; var formObject = new ej.inputs.FormValidator('#form-element', options); formObject.customPlacement = function (element, errorElement) { element.parentNode.parentNode.appendChild(errorElement); }; |

PDF Viewer

Getting Started

Getting Started with ASP.NET MVC Standalone PDF Viewer Control

The [ASP.NET MVC PDF Viewer](#) control is used to viewing and printing PDF files in any web application. It provides the best viewing experience available with core interactions such as zooming, scrolling, text searching, text selection, and text copying. Thumbnail, bookmark, hyperlink and table of contents support provides easy navigation within and outside the PDF files.

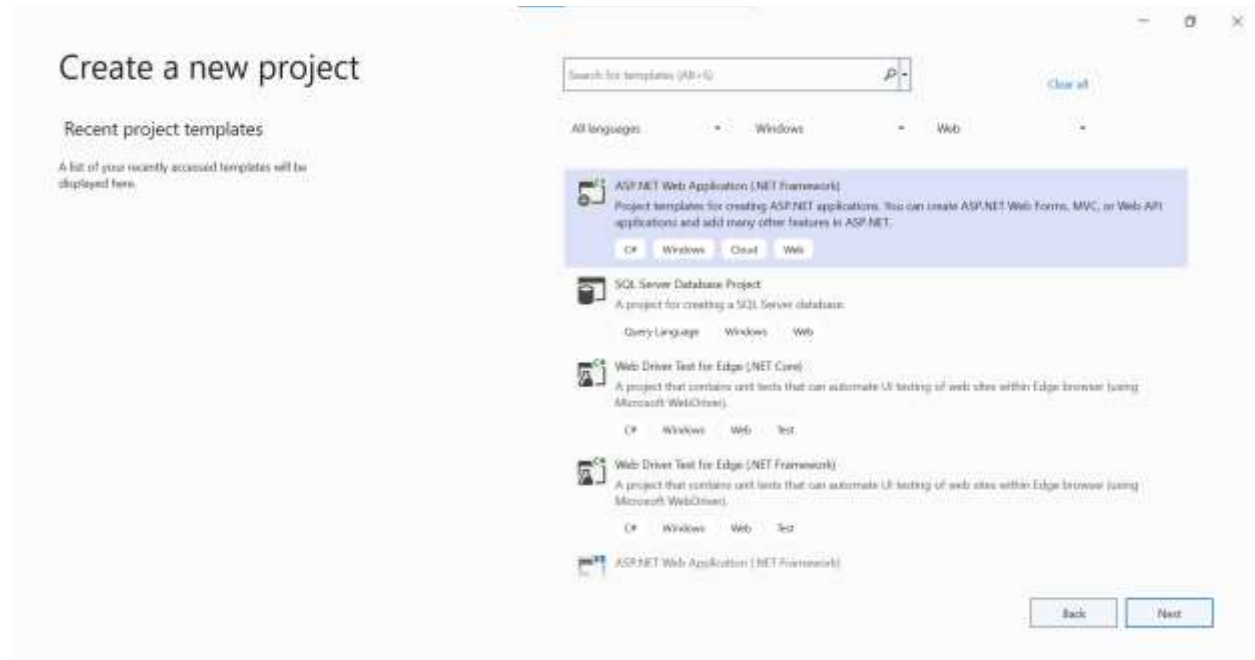
This section briefly explains about how to integrate ASP.NET MVC PDF Viewer control in your ASP.NET MVC application using Visual Studio.

Prerequisites

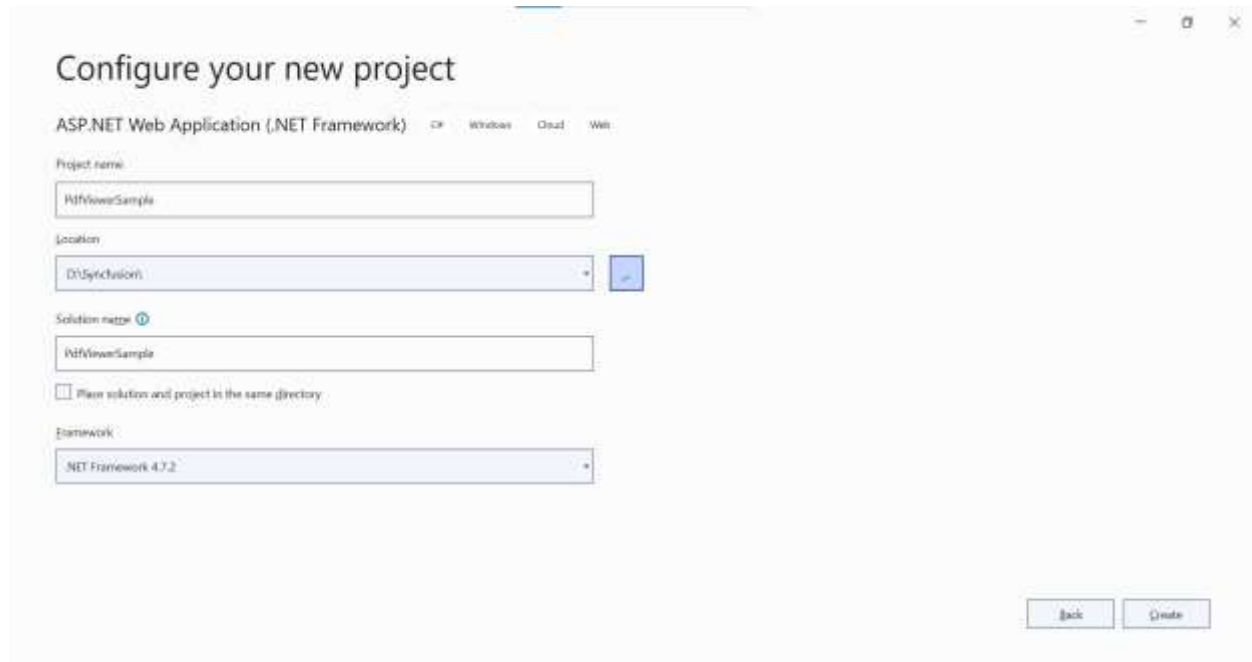
[System requirements for ASP.NET MVC controls](#)

Integrate PDF Viewer into an ASP.NET MVC application

1. Start Visual Studio and select **Create a new project**.
2. Create a new ASP.NET MVC Web Application project.



3. Choose the target framework.



Configure your new project

ASP.NET Web Application (.NET Framework) .NET Framework Windows Cloud Web

Project name
PdfViewerSample

Location
D:\Syncfusion\...

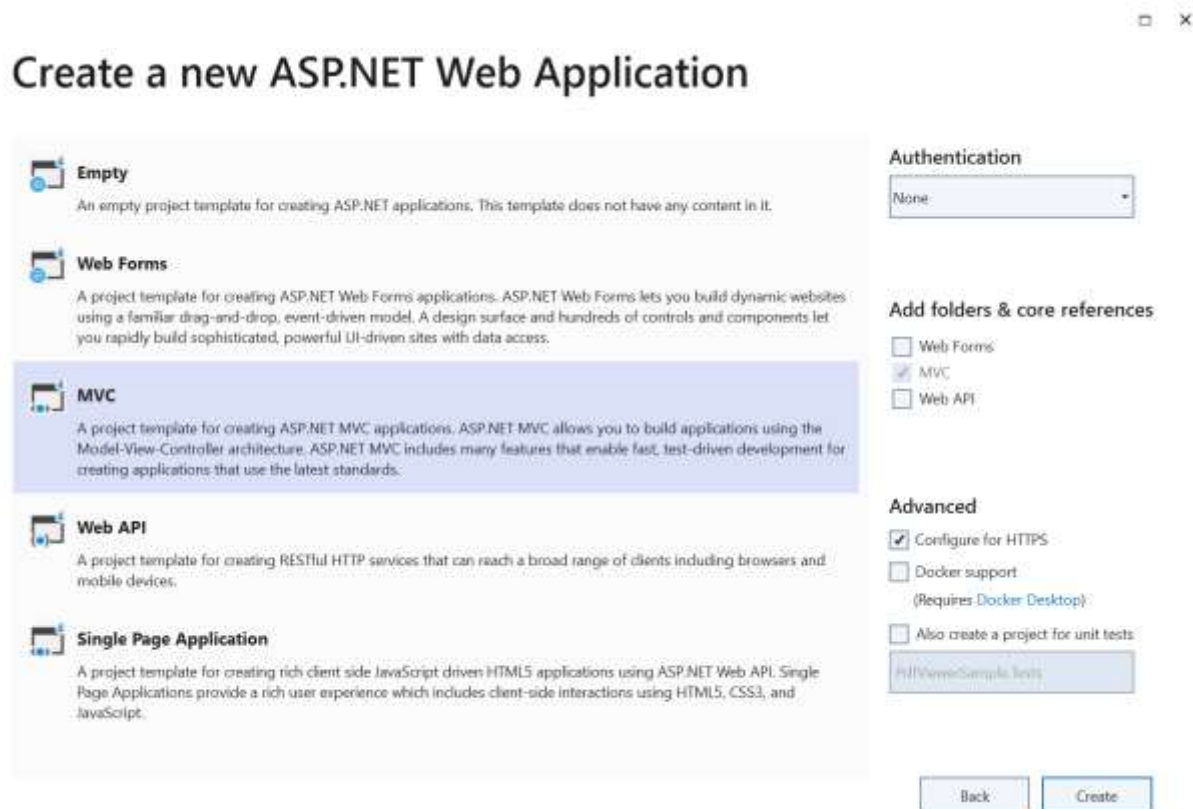
Solution name
PdfViewerSample

☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Back Create

4. Select Web Application pattern (MVC) for the project and then select **Create** button.



Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication
None

Add folders & core references

- ☐ Web Forms
- ☒ MVC
- ☐ Web API

Advanced

- ☒ Configure for HTTPS
- ☐ Docker support
(Requires Docker Desktop)
- ☐ Also create a project for unit tests

Back Create

[ASP.NET MVC PDF Viewer NuGet packages installation](#)

To add .NET PDF Viewer control, the following NuGet packages need to be installed in your ASP.NET MVC application.

- [Syncfusion.EJ2.MVC5](#)

Note: If you creating ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add style sheet

The theme is referred using CDN inside the **<head>** of **~/Views/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls.

Add script reference

Add the required scripts using CDN inside the **<head>** of **~/Views/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Note: While referring the scripts from the downloaded resources in your application, make sure to place the 'ej2-pdfviewer-lib' assets in the same directory as the 'ej2.min.js' script.

Register Syncfusion Script Manager

Open **~/Views/Shared/_Layout.cshtml** page and register the script manager in the ASP.NET MVC application as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Note: Add the script manager `EJS().ScriptManager()` at the end of `<body>`.

Add ASP.NET MVC PDF Viewer control

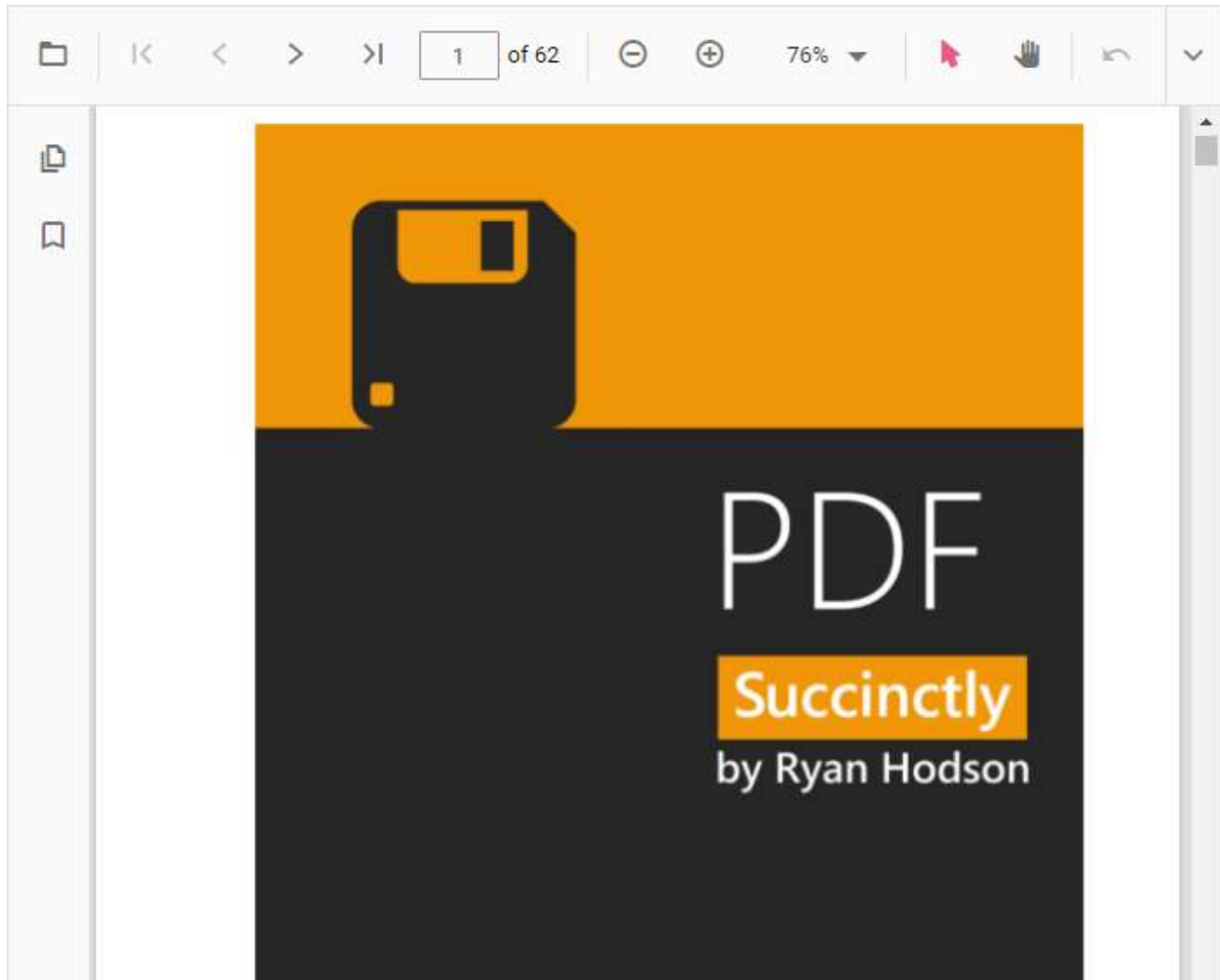
Add the Syncfusion ASP.NET MVC PDF Viewer control in `~/Views/Home/Index.cshtml` page. You can load a PDF file in the PDF Viewer by specifying the document name in the DocumentPath property as below.

~/INDEX.CSHTML

```
@{
    ViewBag.Title = "Home Page";
}
<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
</div>
```

[DocumentPath](#) is the property needed to load a PDF file in the PDF Viewer.

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC PDF Viewer control will be rendered in the default web browser.



Note: [View Sample in GitHub.](#)

Note: You can refer to our [ASP.NET MVC PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC PDF Viewer example](#) to understand the core features of PDF Viewer.

[Limitation over Server-Backed PDF Viewer to Standalone PDF Viewer control](#)

When comparing a Standalone PDF Viewer to a Server-Backed PDF Viewer control, it's crucial to understand the limitations that the Standalone PDF Viewer may have in comparison. These limitations are important to consider

[PNG Image Support](#)

The Standalone PDF Viewer does not have the capability to utilize PNG format for adding images to handwritten annotations, custom stamp, signature and initial form fields. It's important to be aware that only certain image formats, such as JPEG, are compatible for these purposes.

[Local File Access](#)

- The Standalone PDF Viewer control does not have the capability to directly access and load local physical files from a user's device. As a result, it is not possible to use a documentPath to load a PDF file directly from a local server within the viewer.

- The Standalone PDF Viewer allows users to export annotations and form fields from the viewer, it's important to be aware that the viewer does not support the direct import of annotations and form fields from a locally specified file path. In other words, you can extract annotations and form fields from the viewer, but you cannot reintroduce them into the viewer from external sources by specifying a file path located on your local device.

Note: These limitations are temporary and are expected to be addressed in the near future.

Getting Started with ASP.NET MVC PDF Viewer Control

The [ASP.NET MVC PDF Viewer](#) control is used to viewing and printing PDF files in any web application. It provides the best viewing experience available with core interactions such as zooming, scrolling, text searching, text selection, and text copying. Thumbnail, bookmark, hyperlink and table of contents support provides easy navigation within and outside the PDF files.

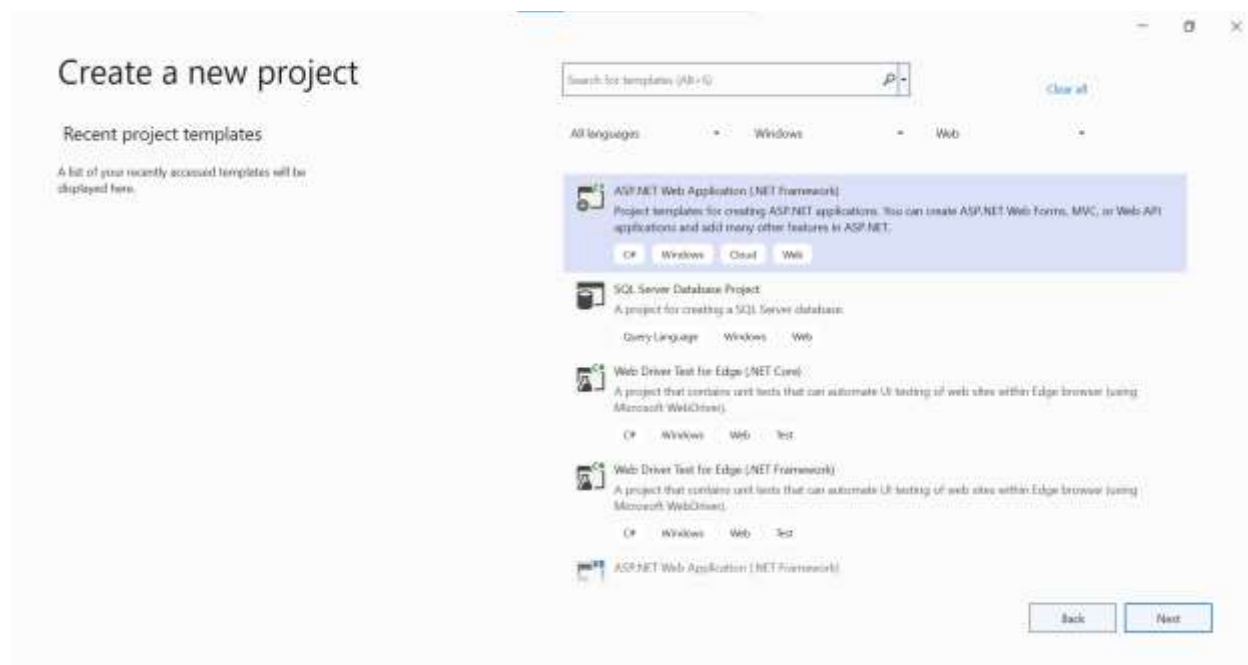
This section briefly explains about how to integrate ASP.NET MVC PDF Viewer control in your ASP.NET MVC application using Visual Studio.

Prerequisites

[System requirements for ASP.NET MVC controls](#)

Integrate PDF Viewer into an ASP.NET MVC application

1. Start Visual Studio and select **Create a new project**.
2. Create a new ASP.NET MVC Web Application project.



3. Choose the target framework.

Configure your new project

ASP.NET Web Application (.NET Framework) CF Windows Cloud Web

Project name: PdfViewerSample

Location: D:\Syncfusion\...

Solution name: PdfViewerSample

☐ Place solution and project in the same directory

Framework: .NET Framework 4.7.2

Back Create

4. Select Web Application pattern (MVC) for the project and then select **Create** button.

Create a new ASP.NET Web Application

Empty
An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms
A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC
A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API
A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application
A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication
None

Add folders & core references

- ☐ Web Forms
- ☒ MVC
- ☐ Web API

Advanced

- ☒ Configure for HTTPS
- ☐ Docker support
(Requires [Docker Desktop](#))
- ☐ Also create a project for unit tests

PdfViewerSample.Tests

Back Create

[ASP.NET MVC PDF Viewer NuGet packages installation](#)

To add .NET PDF Viewer control, the following NuGet packages need to be installed in your ASP.NET MVC application.

- [Syncfusion.EJ2.PdfViewer.AspNet.Mvc5](#)
- [Syncfusion.EJ2.MVC5](#)

Note: If you creating ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

Add style sheet

The theme is referred using CDN inside the **<head>** of **~/Views/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
</head>
```

Note: Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls.

Add script reference

Add the required scripts using CDN inside the **<head>** of **~/Views/Shared/_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

Register Syncfusion Script Manager

Open **~/Views/Shared/_Layout.cshtml** page and register the script manager in the ASP.NET MVC application as follows.

~/ LAYOUT.CSHTML

```
<body>
...
```

```
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Note: Add the script manager `EJS().ScriptManager()` at the end of `<body>`.

Add ASP.NET MVC PDF Viewer control

Add the Syncfusion ASP.NET MVC PDF Viewer control in `~/Views/Home/Index.cshtml` page. You can load a PDF file in the PDF Viewer by specifying the document name in the DocumentPath property as below.

~/INDEX.CSHTML

```
@{
    ViewBag.Title = "Home Page";
}
<div>
<div style="height:500px;width:100%;">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/Home/").DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
</div>
```

Add the below code in the `HomeController.cs` file which is placed inside `Controllers` folder.

~/HOMECONTROLLER.CS

```
using Newtonsoft.Json;
using Syncfusion.EJ2.PdfViewer;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Reflection;
using System.Web;
using System.Web.Mvc;
namespace GettingStartedMVC.Controllers
{
    public class HomeController : Controller
    {
        [System.Web.Mvc.HttpPost]
        public ActionResult Load(jsonObjects jsonObject)
        {
            PdfRenderer pdfviewer = new PdfRenderer();
            MemoryStream stream = new MemoryStream();
            var jsonData = JsonConvert.DeserializeObject(jsonObject);
            object jsonResult = new object();
            if (jsonObject != null && jsonData.ContainsKey("document"))
            {
                if (bool.Parse(jsonData["isFileName"]))
                {
                    string documentPath = GetDocumentPath(jsonData["document"]);
```

```

if (!string.IsNullOrEmpty(documentPath))
{
    byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
    stream = new MemoryStream(bytes);
}
else
{
    string fileName = jsonData["document"].Split(new string[] { "://" },
    StringSplitOptions.None)[0];
    if (fileName == "http" || fileName == "https")
    {
        var WebClient = new WebClient();
        byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]);
        stream = new MemoryStream(pdfDoc);
    }
    else
    {
        return this.Content(jsonData["document"] + " is not found");
    }
}
else
{
    byte[] bytes = Convert.FromBase64String(jsonData["document"]);
    stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}
public Dictionary<string, string> JsonConverter(jsonObjects results)
{
    Dictionary<string, object> resultObjects = new Dictionary<string, object>();
    resultObjects = results.GetType().GetProperties(BindingFlags.Instance |
    BindingFlags.Public)
    .ToDictionary(prop => prop.Name, prop => prop.GetValue(results, null));
    var emptyObjects = (from kv in resultObjects
    where kv.Value != null
    select kv).ToDictionary(kv => kv.Key, kv => kv.Value);
    Dictionary<string, string> jsonResult = emptyObjects.ToDictionary(k =>
    k.Key, k => k.Value.ToString());
    return jsonResult;
}
[System.Web.Mvc.HttpPost]
public ActionResult ExportAnnotations(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    string jsonResult = pdfviewer.ExportAnnotation(jsonData);
    return Content((jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult ImportAnnotations(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    string jsonResult = string.Empty;
    var jsonData = JsonConverter(jsonObject);

```

```
if (jsonObject != null && jsonData.ContainsKey("fileName"))
{
    string documentPath = GetDocumentPath(jsonData["fileName"]);
    if (!string.IsNullOrEmpty(documentPath))
    {
        jsonResult = System.IO.File.ReadAllText(documentPath);
    }
    else
    {
        return this.Content(jsonData["document"] + " is not found");
    }
}
return Content(JsonConvert.SerializeObject(jsonResult));
[System.Web.Mvc.HttpPost]
public ActionResult ImportFormFields(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    object jsonResult = pdfviewer.ImportFormFields(jsonData);
    return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult ExportFormFields(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    string jsonResult = pdfviewer.ExportFormFields(jsonData);
    return Content(jsonResult);
}
[System.Web.Mvc.HttpPost]
public ActionResult RenderPdfPages(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    object jsonResult = pdfviewer.GetPage(jsonData);
    return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult Unload(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    pdfviewer.ClearCache(jsonData);
    return this.Content("Document cache is cleared");
}
[System.Web.Mvc.HttpPost]
public ActionResult RenderThumbnailImages(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    object result = pdfviewer.GetThumbnailImages(jsonData);
    return Content(JsonConvert.SerializeObject(result));
}
[System.Web.Mvc.HttpPost]
public ActionResult Bookmarks(jsonObjects jsonObject)
{

```

```

PdfRenderer pdfviewer = new PdfRenderer();
var jsonData = JsonConvert.SerializeObject(jsonObject);
object jsonResult = pdfviewer.GetBookmarks(jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult RenderAnnotationComments(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConvert.SerializeObject(jsonObject);
    object jsonResult = pdfviewer.GetAnnotationComments(jsonData);
    return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult Download(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConvert.SerializeObject(jsonObject);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
    return Content(documentBase);
}
[System.Web.Mvc.HttpPost]
public ActionResult PrintImages(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConvert.SerializeObject(jsonObject);
    object pageImage = pdfviewer.GetPrintImage(jsonData);
    return Content(JsonConvert.SerializeObject(pageImage));
}
private HttpResponseMessage GetPlainText(string pageImage)
{
    var responseText = new HttpResponseMessage(HttpStatusCode.OK);
    responseText.Content = new StringContent(pageImage,
        System.Text.Encoding.UTF8, "text/plain");
    return responseText;
}
private string GetDocumentPath(string document)
{
    string documentPath = string.Empty;
    if (!System.IO.File.Exists(document))
    {
        var path = HttpContext.Request.PhysicalApplicationPath;
        if (System.IO.File.Exists(path + "App_Data\\" + document))
            documentPath = path + "App_Data\\" + document;
    }
    else
    {
        documentPath = document;
    }
    return documentPath;
}
public ActionResult Index()
{
    return View();
}
public ActionResult About()
{

```

```

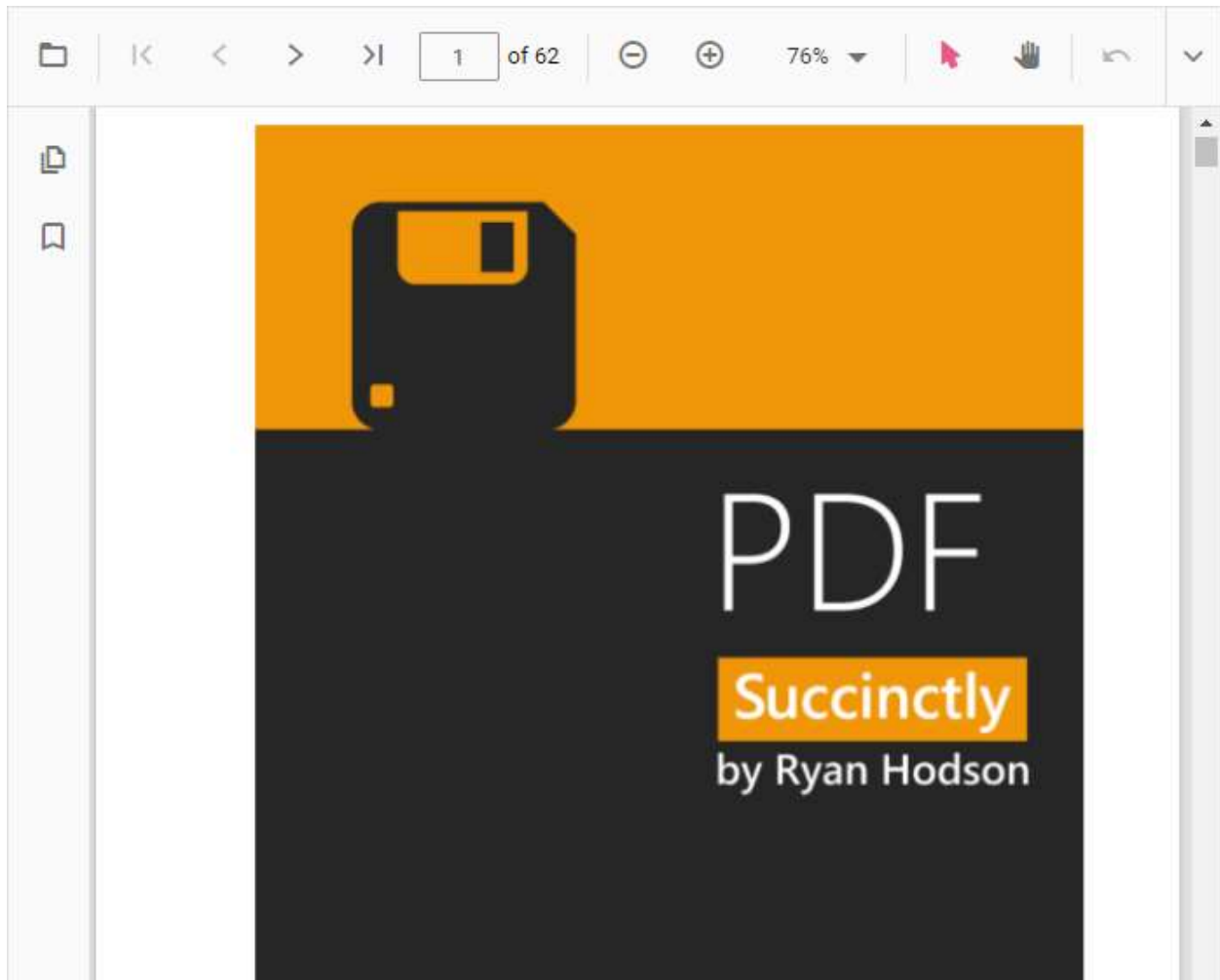
ViewBag.Message = "Your application description page.";
return View();
}
public ActionResult Contact()
{
    ViewBag.Message = "Your contact page.";
    return View();
}
}
public class jsonObjects
{
    public string document { get; set; }
    public string password { get; set; }
    public string zoomFactor { get; set; }
    public string isFileName { get; set; }
    public string xCoordinate { get; set; }
    public string yCoordinate { get; set; }
    public string pageNumber { get; set; }
    public string documentId { get; set; }
    public string hashId { get; set; }
    public string sizeX { get; set; }
    public string sizeY { get; set; }
    public string startPage { get; set; }
    public string endPage { get; set; }
    public string stampAnnotations { get; set; }
    public string textMarkupAnnotations { get; set; }
    public string stickyNotesAnnotation { get; set; }
    public string shapeAnnotations { get; set; }
    public string measureShapeAnnotations { get; set; }
    public string action { get; set; }
    public string pageStartIndex { get; set; }
    public string pageEndIndex { get; set; }
    public string fileName { get; set; }
    public string elementId { get; set; }
    public string pdfAnnotation { get; set; }
    public string importPageList { get; set; }
    public string uniqueId { get; set; }
    public string data { get; set; }
    public string viewPortWidth { get; set; }
    public string viewportHeight { get; set; }
    public string tilecount { get; set; }
    public string isCompletePageSizeNotReceived { get; set; }
    public string freeTextAnnotation { get; set; }
    public string signatureData { get; set; }
    public string fieldsData { get; set; }
    public string FormDesigner { get; set; }
    public string inkSignatureData { get; set; }
}
}

```

[ServiceUrl](#) is necessary to communicate with the server which also specifies the path of the controller. Here, PdfViewer is the name of the controller.

[DocumentPath](#) is the property needed to load a PDF file in the PDF Viewer.

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC PDF Viewer control will be rendered in the default web browser.



Note: We have provided the support to dynamically change the `serviceURL`. So, after changing the `serviceURL` dynamically, you need invoke the `pdfViewer.dataBind()` method to update the `serviceURL` quickly. This will effectively change the `serviceURL` dynamically. Ensure that this step is performed after version 23.1.36.

```
string serviceUrl = VirtualPathUtility.ToAbsolute("~/Home/");  
function load() {  
    var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];  
    pdfViewer.serviceUrl = '@serviceUrl'  
    pdfViewer.documentPath = "https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf";  
    pdfViewer.dataBind();  
    pdfViewer.load(pdfViewer.documentPath, null);  
}
```

Note: [View Sample in GitHub.](#)

Note: You can refer to our [ASP.NET MVC PDF Viewer](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC PDF Viewer example](#) to understand the core features of PDF Viewer.

Note: When configuring the server-backed PDF viewer, it's essential to understand that there is no need to include the pdfium.js and pdfium.wasm files. Unlike the standalone PDF viewer, which relies on these files for local rendering, the server-backed PDF viewer fetches and renders PDFs directly from the server. Consequently, you can exclude the copy command for deployment process, as they are not required to load and display PDFs in this context.

Feature modules

The PDF Viewer features are segregated into individual feature-wise modules to enable selectively referencing in the application. The required modules should be injected to extend its functionality. The following are the selective modules of PDF Viewer that can be included as required:

The available PdfViewer modules are:

- **Toolbar**:- Built in toolbar for better user interaction.
- **Magnification**:- Perform zooming operation for better viewing experience.
- **Navigation**:- Easy navigation across the PDF pages.
- **LinkAnnotation**:- Easy navigation within and outside of the PDF document.
- **ThumbnailView**:- Easy navigation with in the PDF document.
- **BookmarkView**:- Easy navigation based on the bookmark content of the PDF document.
- **TextSelection**:- Select and copy text from a PDF file.
- **TextSearch**:- Search a text easily across the PDF document.
- **Print**:- Print the entire document or a specific page directly from the browser.
- **Annotation**:- Annotations can be added or edited in the PDF document.

Note: In addition to injecting the required modules in your application, enable corresponding properties to extend the functionality for a PDF Viewer instance.

Refer to the following table.

| Module | Property to enable the functionality for a PDF Viewer instance |
|----------------|--|
| --- | --- |
| Toolbar | <code>@Html.EJS().PdfViewer("container").EnableToolbar(true).Render()</code> |
| Magnification | <code>@Html.EJS().PdfViewer("container").EnableMagnification(true).Render()</code> |
| Navigation | <code>@Html.EJS().PdfViewer("container").EnableNavigation(true).Render()</code> |
| LinkAnnotation | <code>@Html.EJS().PdfViewer("container").EnableHyperlink(true).Render()</code> |
| ThumbnailView | <code>@Html.EJS().PdfViewer("container").EnableThumbnail(true).Render()</code> |
| BookmarkView | <code>@Html.EJS().PdfViewer("container").EnableBookmark(true).Render()</code> |
| TextSelection | <code>@Html.EJS().PdfViewer("container").EnableTextSelection(true).Render()</code> |
| TextSearch | <code>@Html.EJS().PdfViewer("container").EnableTextSearch(true).Render()</code> |


```
|Print|@Html.EJS().PdfViewer("container").EnablePrint(true).Render()|
```

```
|Annotation|@Html.EJS().PdfViewer("container").EnableAnnotation(true).Render()|
```

See also

- [Toolbar items](#)
- [Toolbar customization](#)

Open PDF files

Open PDF file from AWS S3

To load a PDF file from AWS S3 in a PDF Viewer, you can follow the steps below

Step 1: Create AWS S3 account

Set up an AWS S3 account by following the instructions on the official AWS site: [AWS Management Console](#). Create an S3 bucket and generate access keys while ensuring secure storage of credentials.

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the HomeController.cs File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
`
```

2. Modify the `Load()` method to load the PDF files from AWS S3.

```
`csharp
private readonly string _accessKey = "Your Access Key from AWS S3";
private readonly string _secretKey = "Your Secret Key from AWS S3";
private readonly string _bucketName = "Your Bucket name from AWS S3";
[System.Web.Mvc.HttpPost]
public async Task<ActionResult> Load(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    MemoryStream stream = new MemoryStream();
```

```

var jsonData = JsonConvert(jsonObject);
object jsonResult = new object();
if (jsonObject != null && jsonData.ContainsKey("document"))
{
    if (bool.Parse(jsonData["isFileName"]))
    {
        RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        // Configure the AWS SDK with your access credentials and other settings
        var s3Client = new AmazonS3Client(accessKey, secretKey, bucketRegion);
        string document = jsonData["document"];
        // Specify the document name or retrieve it from a different source
        var response = await s3Client.GetObjectAsync(_bucketName, document);
        Stream responseStream = response.ResponseStream;
        responseStream.CopyTo(stream);
        stream.Seek(0, SeekOrigin.Begin);
    }
    else
    {
        byte[] bytes = Convert.FromBase64String(jsonData["document"]);
        stream = new MemoryStream(bytes);
    }
}
jsonResult = pdfviewer.Load(stream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}
`

```

Note: Replace **Your Access Key from AWS S3**, **Your Secret Key from AWS S3**, and **Your Bucket name from AWS S3** with your actual AWS access key, secret key and bucket name

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from AWS S3. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```

`csharp
@{

```

```

ViewBag.Title = "Home Page";
}
<div>
<div style="height:500px;width:100%;">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`

```

Note: The **AWSSDK.S3** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub](#)

Open PDF file from Azure Blob Storage

To load a PDF file from Azure Blob Storage in a PDF Viewer, you can follow the steps below

Step 1: Create the Azure Blob Storage account

Log in to the Azure Portal. Create a new Storage Account with preferred settings. Note access keys during the setup. Within the Storage Account, create a Blob Container following the steps in this [link](#).

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```

`csharp
using System.IO;
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Specialized;
`

```

2. Modify the `Load()` method to load the PDF files from Azure Blob Storage.

```

`csharp
private readonly string _connectionString = "Your Connection string from Azure";
private readonly string _containerName = "Your container name in Azure";
public ActionResult Load(jsonObjects jsonObject)
{

```

```

PdfRenderer pdfviewer = new PdfRenderer();
MemoryStream stream = new MemoryStream();
var jsonData = JsonConvert(jsonObject);
object jsonResult = new object();
if (jsonObject != null && jsonData.ContainsKey("document"))
{
    if (bool.Parse(jsonData["isFileName"]))
    {
        // Create a BlobServiceClient object by passing the connection string.
        BlobServiceClient blobServiceClient = new BlobServiceClient(_connectionString);
        // Get a reference to the container
        BlobContainerClient containerClient = blobServiceClient.GetBlobContainerClient(_containerName);
        string fileName = jsonData["document"];
        // Get a reference to the block blob
        BlockBlobClient blockBlobClient = containerClient.GetBlockBlobClient(fileName);
        blockBlobClient.DownloadTo(stream);
    }
    else
    {
        byte[] bytes = Convert.FromBase64String(jsonData["document"]);
        stream = new MemoryStream(bytes);
    }
}
jsonResult = pdfviewer.Load(stream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}
`

```

Note: Replace **Your Connection string from Azure** with the actual connection string for your Azure Blob Storage account and **Your container name in Azure** with the actual container name

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Azure Blob Storage. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

`csharp

```
@{
    ViewBag.Title = "Home Page";
}

<div>

<div style="height:500px;width:100%;">

@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()

</div>

</div>
,
```

Note: The **Azure.Storage.Blobs** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub.](#)

Open PDF file from Google Cloud Storage

To load a PDF file from Google Cloud Storage in a PDF Viewer, you can follow the steps below

Step 1 Create a Service Account

Open the Google Cloud Console. Navigate to **IAM & Admin > Service accounts**. Click **Create Service Account**. Enter a name, assign roles (e.g., Storage Object Admin), and create a key in JSON format. Download the key file securely. Utilize the downloaded key file in your applications or services for authentication and access to the Google Cloud Storage bucket. For additional details, refer to the [official documentation](#).

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the **HomeController.cs** File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Google.Cloud.Storage.V1;
using Google.Apis.Auth.OAuth2;
,
```

2. Add the following private fields and constructor parameters to the **HomeController.cs** class, In the constructor, assign the values from the configuration to the corresponding fields

```
`csharp
```

```
// The key file is used to authenticate with Google Cloud Storage.
private string keyFilePath = @"path/to/service-account-key.json";
private readonly string _bucketName = "Your Bucket name from Google Cloud Storage";
private readonly StorageClient _storageClient;
public HomeController()
{
    // Load the service account credentials from the key file.
    var credentials = GoogleCredential.FromFile(keyFilePath);
    // Create a storage client with Application Default Credentials
    _storageClient = StorageClient.Create(credentials);
}
`
```

3. Modify the `Load()` method to load the PDF files from Google Cloud Storage bucket.

```
`csharp
public ActionResult Load(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    MemoryStream stream = new MemoryStream();
    var jsonData = JsonConvert.DeserializeObject(jsonObject);
    object jsonResult = new object();
    if (jsonObject != null && jsonData.ContainsKey("document"))
    {
        if (bool.Parse(jsonData["isFileName"]))
        {
            string bucketName = _bucketName;
            string fileName = jsonData["document"];
            _storageClient.DownloadObject(bucketName, fileName, stream);
            stream.Position = 0;
        }
        else
        {
            byte[] bytes = Convert.FromBase64String(jsonData["document"]);
        }
    }
}
```

```

stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}
`

```

Note: Replace **Your Bucket name from Google Cloud Storage** with the actual name of your Google Cloud Storage bucket

Note: Replace **path/to/service-account-key.json** with the actual file path to your service account key JSON file. Make sure to provide the correct path and filename.

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Google Cloud Storage. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```

`csharp
@{
    ViewBag.Title = "Home Page";
}
<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`

```

Note: The **Google.Cloud.Storage.V1** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub](#)

Open PDF file from Google Drive

To load a PDF file from Google Drive in a PDF Viewer, you can follow the steps below

Step 1 Set up Google Drive API

You must set up a project in the Google Developers Console and enable the Google Drive API. Obtain the necessary credentials to access the API. For more information, view the official [link](#).

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Google.Apis.Drive.v3;
using Google.Apis.Auth.OAuth2;
using Google.Apis.Services;
using Google.Apis.Util.Store;
`
```

2. Modify the `Load()` method to load the PDF files from Google Drive.

```
`csharp
// Specify the path to the credentials file
private string credentialsPath = "Your Path to the OAuth 2.0 Client IDs json file";
// Specify the folder ID where you want to upload the PDF on Google Drive
private readonly string folderId = "Your Google Drive Folder ID";
private readonly string ApplicationName = "Your Application Name";
private readonly string tokenjson = "Path to create token.json file";
public async Task<ActionResult> Load(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    MemoryStream stream = new MemoryStream();
    var jsonData = JsonConvert.DeserializeObject(jsonObject);
    object jsonResult = new object();
    if (jsonObject != null && jsonData.ContainsKey("document"))
    {
        if (bool.Parse(jsonData["isFileName"]))
        {
            string objectName = jsonData["document"];
            // Google Drive API setup
            UserCredential credential;
```



```
string[] Scopes = { DriveService.Scope.DriveReadOnly };
// Load the credentials file
using (var streammen = new FileStream(credentialsPath, FileMode.Open, FileAccess.Read))
{
    // The file token.json stores the user's access and refresh tokens, and is created
    // automatically when the authorization flow completes for the first time.
    string tokenPath = tokenjson;
    credential = GoogleWebAuthorizationBroker.AuthorizeAsync(
        GoogleClientSecrets.Load(streammen).Secrets,
        Scopes,
        "user",
        CancellationToken.None,
        new FileDataStore(tokenPath, true)).Result;
}
// Create the Drive service
var service = new DriveService(new BaseClientService.Initializer()
{
    HttpClientInitializer = credential,
    ApplicationName = ApplicationName,
});
FilesResource.ListRequest listRequest = service.Files.List();
listRequest.Q = "mimeType='application/pdf' and '" + folderId + "' in parents and trashed=false";
listRequest.Fields = "files(id, name)";
var files = await listRequest.ExecuteAsync();
// Process the list of files (you can use 'files' to retrieve the list of files)
string fileIdToDownload = null;
foreach (var file in files.Files)
{
    string fileId = file.Id;
    string fileName = file.Name;
    if (fileName == objectName)
    {
        // Save the matching fileId
    }
}
```

```

fileIdToDownload = fileId;
break;
}
}
string fileIds = fileIdToDownload;
var request = service.Files.Get(fileIds);
await request.DownloadAsync(stream);
stream.Position = 0;
}
else
{
byte[] bytes = Convert.FromBase64String(jsonData["document"]);
stream = new MemoryStream(bytes);
}
}
jsonResult = pdfviewer.Load(stream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}
,

```

Note: Replace **Your Google Drive Folder ID**, **Your Application name**, **tokenPath** and **Your Path to the OAuth 2.0 Client IDs json file** with your actual Google drive folder ID , Your name for your application, Path to the token file to generate and the path for the JSON file.

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Google Drive. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```

`csharp
@{
ViewBag.Title = "Home Page";
}
<div>
<div style="height:500px;width:100%;">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>

```

```
</div>
```

Note: The **Google.Apis.Drive.v3** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub](#)

Open PDF file from Box cloud file storage

To load a PDF file from Box cloud file storage in a PDF Viewer, you can follow the steps below

Step 1 Set up a Box developer account and create a Box application

To access Box storage programmatically, you'll need a developer account with Box. Go to the [Box Developer Console](#), sign in or create a new account, and then create a new Box application. This application will provide you with the necessary credentials Client ID and Client Secret to authenticate and access Box APIs. Before accessing files, you need to authenticate your application to access your Box account. Box API supports **OAuth 2.0 authentication** for this purpose.

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the **HomeController.cs** File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using Box.V2;
using Box.V2.Auth;
using Box.V2.Config;
using Box.V2.Models;
`
```

2. Modify the **Load()** method to load the PDF files from Box cloud file storage.

```
`csharp
private readonly string clientId = "YourBoxStorageClientID";
private readonly string clientSecret = "YourBoxStorageClientSecret";
private readonly string accessToken = "YourBoxStorageAccess_Token";
private readonly string folderId = "YourFolder_ID";
[System.Web.Mvc.HttpPost]
public async Task<ActionResult> Load(jsonObjects jsonObject)
{
```

```
PdfRenderer pdfviewer = new PdfRenderer();
MemoryStream stream = new MemoryStream();
var jsonData = JsonConvert(jsonObject);
object jsonResult = new object();
if (jsonObject != null && jsonData.ContainsKey("document"))
{
    if (bool.Parse(jsonData["isFileName"]))
    {
        string objectName = jsonData["document"];
        // Initialize the Box API client with your authentication credentials
        var auth = new OAuthSession(accessToken, "YOURREFRESH_TOKEN", 3600, "bearer");
        var config = new BoxConfigBuilder(clientId, clientSecret, new Uri("http://boxsdk")).Build();
        var client = new BoxClient(config, auth);
        // Download the file from Box storage
        var items = await client.FoldersManager.GetFolderItemsAsync(_folderId, 1000, autoPaginate: true);
        var files = items.Entries.Where(i => i.Type == "file");
        // Filter the files based on the objectName
        var matchingFile = files.FirstOrDefault(file => file.Name == objectName);
        // Fetch the file from Box storage by its name
        var fileStream = await client.FilesManager.DownloadAsync(matchingFile.Id);
        stream = new MemoryStream();
        await fileStream.CopyToAsync(stream);
        // Reset the position to the beginning of the stream
        stream.Position = 0;
    }
    else
    {
        byte[] bytes = Convert.FromBase64String(jsonData["document"]);
        stream = new MemoryStream(bytes);
    }
}
jsonResult = pdfviewer.Load(stream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
```

```
}
`
```

Note: replace **YourBoxStorageAccessToken** with your actual box access token, and **YourFolderID** with the ID of the folder in your box storage where you want to perform specific operations. Remember to use your valid box API credentials, as **YourBoxStorageClientID and YourBoxStorageClientSecret** are placeholders for your application's API key and secret.

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Box cloud file storage. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```
`csharp
@{
    ViewBag.Title = "Home Page";
}

<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`
```

Note: The **Box.V2.Core** NuGet package must be installed in your application to use the previous code example.

Note: Replace `PDF_Succinctly.pdf` with the actual document name that you want to load from Box cloud file storage. Make sure to pass the document name from the box folder to the `documentPath` property of the PDF viewer component

[View sample in GitHub](#)

[Save PDF files](#)

[Save PDF file to AWS S3](#)

To save a PDF file to AWS S3 bucket, you can follow the steps below

Step 1: Create AWS S3 account

Set up an AWS S3 account by following the instructions on the official AWS site: [AWS Management Console](#). Create an S3 bucket and generate access keys while ensuring secure storage of credentials.

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model
`
```

3. Modify the `Download()` method to save the downloaded PDF files to AWS S3 container

```
`csharp
private readonly string _accessKey = "Your Access Key from AWS S3";
private readonly string _secretKey = "Your Secret Key from AWS S3";
private readonly string _bucketName = "Your Bucket name from AWS S3";
[System.Web.Mvc.HttpPost]
public ActionResult Download(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
    RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
    // Configure the AWS SDK with your access credentials and other settings
    var s3Client = new AmazonS3Client(accessKey, secretKey, bucketRegion);
    string bucketName = _bucketName;
    string documentName = jsonData["documentId"];
    string result = Path.GetFileNameWithoutExtension(documentName);
    byte[] bytes = Convert.FromBase64String(documentBase.Split(',')[1]);
    using (MemoryStream stream = new MemoryStream(bytes))
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = result + "_downloaded.pdf",
            InputStream = stream,

```

```
};
// Upload the PDF document to AWS S3
var response = s3Client.PutObjectAsync(request).Result;
}
return Content(documentBase);
}
`
```

Note: Replace **Your Access Key from AWS S3**, **Your Secret Key from AWS S3**, and **Your Bucket name from AWS S3** with your actual AWS access key, secret key and bucket name

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from AWS S3. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```
`csharp
@{
    ViewBag.Title = "Home Page";
}

<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`
```

Note: The **AWSSDK.S3** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub](#)

Save PDF file to Azure Blob Storage

To save a PDF file to Azure Blob Storage, you can follow the steps below

Step 1: Create the Azure Blob Storage account

Log in to the Azure Portal. Create a new Storage Account with preferred settings. Note access keys during the setup. Within the Storage Account, create a Blob Container following the steps in this [link](#).

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Azure.Storage.Blobs;
using Azure.Storage.Blobs.Specialized;
`
```

3. Modify the `Download()` method to save the downloaded PDF files to Azure Blob Storage container

```
`csharp
private readonly string _connectionString = "Your Connection string from Azure";
private readonly string _containerName = "Your container name in Azure";
public ActionResult Download(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConvert.SerializeObject(jsonObject);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
    string document = jsonData["documentId"];
    // Create a BlobServiceClient object by passing the connection string.
    BlobServiceClient blobServiceClient = new BlobServiceClient(_connectionString);
    // Get a reference to the container
    BlobContainerClient containerClient = blobServiceClient.GetBlobContainerClient(_containerName);
    string result = Path.GetFileNameWithoutExtension(document);
    // Get a reference to the blob
    BlobClient blobClient = containerClient.GetBlobClient(result + "_download.pdf");
    // Convert the document base64 string to bytes
    byte[] bytes = Convert.FromBase64String(documentBase.Split(',')[1]);
    // Upload the document to Azure Blob Storage
    using (MemoryStream stream = new MemoryStream(bytes))
    {
        blobClient.Upload(stream, true);
    }
}
```



```
return Content(documentBase);
}
`
```

Note: Replace **Your Connection string from Azure** with the actual connection string for your Azure Blob Storage account and **Your container name in Azure** with the actual container name

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Azure Blob Storage. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```
`csharp
@{
    ViewBag.Title = "Home Page";
}

<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`
```

Note: The **Azure.Storage.Blobs** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub.](#)

Save PDF file to Google Cloud Storage

To save a PDF file to Google Cloud Storage, you can follow the steps below

Step 1 Create a Service Account

Open the Google Cloud Console. Navigate to **IAM & Admin > Service accounts**. Click **Create Service Account**. Enter a name, assign roles (e.g., Storage Object Admin), and create a key in JSON format. Download the key file securely. Utilize the downloaded key file in your applications or services for authentication and access to the Google Cloud Storage bucket. For additional details, refer to the [official documentation](#).

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Google.Cloud.Storage.V1;
using Google.Apis.Auth.OAuth2;
`
```

2. Add the following private fields and constructor parameters to the `HomeController.cs` class, In the constructor, assign the values from the configuration to the corresponding fields

```
`csharp
// The key file is used to authenticate with Google Cloud Storage.
private string keyFilePath = @"path/to/service-account-key.json";
private readonly string _bucketName = "Your Bucket name from Google Cloud Storage";
private readonly StorageClient _storageClient;
public HomeController()
{
    // Load the service account credentials from the key file.
    var credentials = GoogleCredential.FromFile(keyFilePath);
    // Create a storage client with Application Default Credentials
    _storageClient = StorageClient.Create(credentials);
}
`
```

3. Modify the `Download()` method to save the downloaded PDF files to Google Cloud Storage bucket

```
`csharp
public ActionResult Download(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
    string bucketName = _bucketName;
    string fileName = jsonData["documentId"];
    // Convert the base64 string back to bytes
    string result = Path.GetFileNameWithoutExtension(fileName);
}
```

```
byte[] documentBytes = Convert.FromBase64String(documentBase.Split(',')[1]);
// Upload the document to Google Cloud Storage
using (var memoryStream = new MemoryStream(documentBytes))
{
    storageClient.UploadObject(bucketName, result + "downloaded.pdf", null, memoryStream);
}
return Content(documentBase);
}
```

Note: Replace **Your Bucket name from Google Cloud Storage** with the actual name of your Google Cloud Storage bucket

Note: Replace **path/to/service-account-key.json** with the actual file path to your service account key JSON file. Make sure to provide the correct path and filename.

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Google Cloud Storage. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```
`csharp
@{
    ViewBag.Title = "Home Page";
}

<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
```

Note: The **Google.Cloud.Storage.V1** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub](#)

Save PDF file to Google Drive

To save a PDF file to Google Drive, you can follow the steps below

Step 1 Set up Google Drive API

You must set up a project in the Google Developers Console and enable the Google Drive API. Obtain the necessary credentials to access the API. For more information, view the official [link](#).

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using System.IO;
using Google.Apis.Drive.v3;
using Google.Apis.Auth.OAuth2;
using Google.Apis.Services;
using Google.Apis.Util.Store;
`
```

3. Modify the `Download()` method to save the downloaded PDF files to Google Drive bucket

```
`csharp
// Specify the path to the credentials file
private string credentialsPath = "Your Path to the OAuth 2.0 Client IDs json file";
// Specify the folder ID where you want to upload the PDF on Google Drive
private readonly string folderId = "Your Google Drive Folder ID";
private readonly string ApplicationName = "Your Application Name";
private readonly string tokenjson = "Path to create token.json file";
public async Task<ActionResult> Download(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConverter(jsonObject);
    string[] Scopes = { DriveService.Scope.DriveFile };
    // Download the PDF document
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
    byte[] documentBytes = Convert.FromBase64String(documentBase.Split(',')[1]);
    string documentId = jsonData["documentId"];
    string result = Path.GetFileNameWithoutExtension(documentId);
}
```

```
string fileName = result + "_downloaded.pdf";
UserCredential credential;
using (var streammen = new FileStream(credentialsPath, FileMode.Open, FileAccess.Read))
{
    // The file token.json stores the user's access and refresh tokens and is created
    // automatically when the authorization flow completes for the first time.
    string tokenPath = tokenjson;
    credential = await GoogleWebAuthorizationBroker.AuthorizeAsync(
        GoogleClientSecrets.Load(streammen).Secrets,
        Scopes,
        "user",
        CancellationToken.None,
        new FileDataStore(tokenPath, true));
}
// Create the Drive API service
var service = new DriveService(new BaseClientService.Initializer()
{
    HttpClientInitializer = credential,
    ApplicationName = ApplicationName,
});
// Create file metadata
var fileMetadata = new Google.Apis.Drive.v3.Data.File()
{
    Name = fileName,
    Parents = new List<string> { folderId }
};
FilesResource.CreateMediaUpload request;
// Upload the file to Google Drive
using (var stream = new MemoryStream(documentBytes))
{
    request = service.Files.Create(fileMetadata, stream, "application/pdf");
    request.Fields = "id";
    var uploadedFile = await request.UploadAsync();
}
```

```

}
return Content(documentBase);
}
`

```

Note: Replace **Your Google Drive Folder ID**, **Your Application name**, **tokenPath** and **Your Path to the OAuth 2.0 Client IDs json file** with your actual Google drive folder ID , Your name for your application, Path to the token file to generate and the path for the JSON file.

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Google Drive. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```

`csharp
@{
    ViewBag.Title = "Home Page";
}

<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`

```

Note: The **Google.Apis.Drive.v3** NuGet package must be installed in your application to use the previous code example.

[View sample in GitHub](#)

Save PDF file to Box cloud file storage

To save a PDF file to Box cloud file storage, you can follow the steps below

Step 1 Set up a Box developer account and create a Box application

To access Box storage programmatically, you'll need a developer account with Box. Go to the [Box Developer Console](#), sign in or create a new account, and then create a new Box application. This application will provide you with the necessary credentials Client ID and Client Secret to authenticate and access Box APIs. Before accessing files, you need to authenticate your application to access your Box account. Box API supports **OAuth 2.0 authentication** for this purpose.

Step 2: Create PDF Viewer Sample in ASP.NET MVC

Follow instructions provided in the Syncfusion PDF Viewer Getting Started [Guide](#) to create a simple PDF Viewer sample in ASP.NET MVC.

Step 3: Modify the `HomeController.cs` File in the Project

1. Import the required namespaces at the top of the file.

```
`csharp
using Box.V2;
using Box.V2.Auth;
using Box.V2.Config;
using Box.V2.Models;
`
```

3. Modify the `Download()` method to save the downloaded PDF files to Box cloud file storage folder.

```
`csharp
private readonly string clientId = "YourBoxStorageClientID";
private readonly string clientSecret = "YourBoxStorageClientSecret";
private readonly string accessToken = "YourBoxStorageAccess_Token";
private readonly string folderId = "YourFolder_ID";
public async Task<ActionResult> Download(jsonObjects jsonObject)
{
    PdfRenderer pdfviewer = new PdfRenderer();
    var jsonData = JsonConvert.DeserializeObject(jsonObject);
    string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
    byte[] documentBytes = Convert.FromBase64String(documentBase.Split(',')[1]);
    string documentId = jsonData["documentId"];
    string result = Path.GetFileNameWithoutExtension(documentId);
    string fileName = result + "_downloaded.pdf";
    // Initialize the Box API client with your authentication credentials
    var auth = new OAuthSession(accessToken, "YOURREFRESH_TOKEN", 3600, "bearer");
    var config = new BoxConfigBuilder(clientId, clientSecret, new Uri("http://boxsdk")).Build();
    var client = new BoxClient(config, auth);
    var fileRequest = new BoxFileRequest
    {
        Name = fileName,
        Parent = new BoxFolderRequest { Id = folderId },
    };
}
```

```

using (var stream = new MemoryStream(documentBytes))
{
    var boxFile = await client.FilesManager.UploadAsync(fileRequest, stream);
}
return Content(documentBase);
}
`

```

Note: replace **YourBoxStorageAccessToken** with your actual box access token, and **YourFolderID** with the ID of the folder in your box storage where you want to perform specific operations. Remember to use your valid box API credentials, as **YourBoxStorageClientID and YourBoxStorageClientSecret** are placeholders for your application's API key and secret.

Step 4: Set the PDF Viewer Properties in ASP.NET MVC PDF viewer component

Set the `documentPath` property of the PDF viewer component to the desired name of the PDF file you wish to load from Box cloud file storage. Ensure that you correctly pass the document name from the files available in your azure container to the `documentPath` property.

```

`csharp
@{
    ViewBag.Title = "Home Page";
}

<div>
<div style="height:500px;width:100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
</div>
`

```

Note: The **Box.V2.Core** NuGet package must be installed in your application to use the previous code example.

Note: Replace `PDF_Succinctly.pdf` with the actual document name that you want to load from Box cloud file storage. Make sure to pass the document name from the box folder to the `documentPath` property of the PDF viewer component

[View sample in GitHub](#)

Accessibility in Syncfusion PDF Viewer components

The PDF Viewer component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the PDF Viewer component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support] | |

| [Section 508 Support] | |

| [Screen Reader Support] | |

| [Right-To-Left Support] | |

| [Color Contrast] | |

| [Mobile Device Support] | |

| [Keyboard Navigation Support] | |

| [Accessibility Checker Validation] | |

| [Axe-core Accessibility Validation] | |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

WAI-ARIA attributes

[WAI-ARIA](#) (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the PDF Viewer component:

| Attributes | Purpose |

| --- | --- |

| **aria-disabled** | Indicates whether the PDF Viewer component is in a disabled state or not. |

| **aria-expanded** | Indicates whether the suggestion list has expanded or not. |

| **aria-readonly** | Indicates the readonly state of the PDF Viewer element. |

| **aria-haspopup** | Indicates whether the PDF Viewer input element has a suggestion list or not. |

| **aria-label** | Indicates the breadcrumb item text. |

| **aria-labelledby** | Provides a label for the PDF Viewer. Typically, the "aria-labelledby" attribute will contain the id of the element used as the PDF Viewer's title. |

| **aria-describedby** | This attribute points to the PDF Viewer element describing the one it's set on. |

| **aria-orientation** | Indicates whether the PDF Viewer element is oriented horizontally or vertically. |

| **aria-valuetext** | Returns the current text of the PDF Viewer. |

| **aria-valuemax** | Indicates the Maximum value of the PDF Viewer. |

| **aria-valuemin** | Indicates the Minimum value of the PDF Viewer. |

| **aria-valuenow** | Indicates the current value of the PDF Viewer. |

| **aria-controls** | Attribute is set to the button and it points to the corresponding content. |

Keyboard interaction

The PDF Viewer component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

| **Press (Windows)** | **Press (Macintosh)** | **To do this** |

| --- | --- | --- |

|| **Shortcuts for page navigation** |

| Home | Function + Left arrow | Navigate to the first page |

| End | Function + Right arrow | Navigate to the last page |

| Up Arrow | Up Arrow | Navigate to the previous page |

| Down Arrow | Down Arrow | Navigate to the next page |

|| **Shortcuts for Zooming** |

| CONTROL + = | COMMAND + = | Perform zoom-in operation |

| CONTROL + - | COMMAND + - | Perform zoom-out operation |

| CONTROL + 0 | COMMAND + 0 | Retain the zoom level to 1 |

|| **Shortcut for Text Search** |

| CONTROL + F | COMMAND + F | Open the search toolbar |

|| **Shortcut for Text Selection** |

| CONTROL + C | CONTROL + C | Copy the selected text or annotation or form field |

| CONTROL + X | CONTROL + X | Cut the selected text or annotation of the form field |

| CONTROL + Y | CONTROL + Y | Paste the selected text or annotation or form field |

|| Shortcuts for the general operation |

| CONTROL + Z | CONTROL + Z | Undo the action |

| CONTROL + Y | CONTROL + Y | Redo the action |

| CONTROL + P | CONTROL + P | Print the document |

| Delete | Delete | Delete the annotations and form fields |

Ensuring accessibility

The PDF Viewer component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the PDF Viewer component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the PDF Viewer component with accessibility tools.

Note: Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

Built-in toolbar

The PDF Viewer comes with a powerful built-in toolbar to execute important actions such as page navigation, text search, view mode, download print, bookmark and thumbnails.

The following table shows built-in toolbar items and its actions:-

| Option | Description |
|--------|-------------|
|--------|-------------|

| | |
|-----|-----|
| --- | --- |
|-----|-----|

| | |
|------------|---|
| OpenOption | This option provides an action to load the PDF documents to the PDF Viewer. |
|------------|---|

| | |
|--------------------|--|
| PageNavigationTool | This option provides an action to navigate the pages in the PDF Viewer. It contains GoToFirstPage, GoToLastPage, GotoPage, GoToNext, and GoToLast tools. |
|--------------------|--|

| | |
|-------------------|---|
| MagnificationTool | This option provides an action to magnify the pages either with predefined or user defined zoom factors in the PDF Viewer. Contains ZoomIn, ZoomOut, Zoom, FitPage and FitWidth tools |
|-------------------|---|

| | |
|---------|---|
| PanTool | This option provides an action for panning the pages in the PDF Viewer. |
|---------|---|

| | |
|---------------|--|
| SelectionTool | This option provides an action to enable/disable the text selection in the PDF Viewer. |
|---------------|--|

| | |
|--------------|---|
| SearchOption | This option provides an action to search a word in the PDF documents. |
|--------------|---|

| | |
|-------------|--|
| PrintOption | This option provides an action to print the PDF document being loaded in the PDF Viewer. |
|-------------|--|

| | |
|----------------|--|
| DownloadOption | This Download option provides an action to download the PDF document that has been loaded in the PDF Viewer. |
|----------------|--|

| | |
|--------------|---|
| UndoRedoTool | This tool provides options to undo and redo the annotation actions performed in the PDF Viewer. |
|--------------|---|

| AnnotationEditTool | This tool provides options to enable or disable the edit mode of annotation in the PDF Viewer. |

Show/Hide the default toolbar

The PDF Viewer has an option to show or hide the complete default toolbar. You can achieve this by using the following two ways.,

- **Show/Hide toolbar using enableToolbar API as in the following code snippet.**

STANDALONE

```
`html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableToolbar(false).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
`
```

SERVER-BACKED

```
`html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/")).EnableToolbar(false).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
`
```

- **Show/Hide toolbar using showToolbar as in the following code snippet.**

```
`html
<button id="viewer" onclick="enableToolbar()">EnableToolbar</button>
<script>
function enableToolbar() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.toolbar.showToolbar(true);
}
</script>
`
```

Show/Hide the default toolbaritem

The PDF Viewer has an option to show or hide these grouped items in the default toolbar.

- **Show/Hide toolbaritem using toolbarSettings as in the following code snippet.**

STANDALONE

```

<<<html
<button id="viewer" onclick="enableToolBarItem()">EnableToolBarItem</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableToolBar(true).ToolBarSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerToolBarSettings { ShowTooltip = true,
ToolBarItems = "OpenOption"
}).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
</div>
<script>
function enableToolBarItem() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.toolbar.showToolBarItem(new Array("DownloadOption"), true);
}
</script>
<<<

```

SERVER-BACKED

```

<<<html
<button id="viewer" onclick="enableToolBarItem()">EnableToolBarItem</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).EnableToolBar(false).ToolBarSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerToolBarSettings{ ShowTooltip = true,
ToolBarItem = "OpenOption"
}).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
</div>
<script>
function enableToolBarItem() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.toolbar.showToolBarItem(new Array("DownloadOption"), true);
}
</script>
<<<

```

See also

- [ToolBar customization](#)
- [Feature Modules](#)

Navigation in PDF Viewer component

The ASP.NET Core PDF Viewer supports different internal and external navigations.

ToolBar page navigation option

The default toolbar of PDF Viewer contains the following navigation options:

- **Go to page:-** Navigates to the specific page of a PDF document.
- **Show next page:-** Navigates to next page of a PDF document.
- **Show previous page:-** Navigates to the previous page of a PDF document.
- **Show first page:-** Navigates to the first page of a PDF document.

- **Show last page:-** Navigates to the last page of a PDF document.

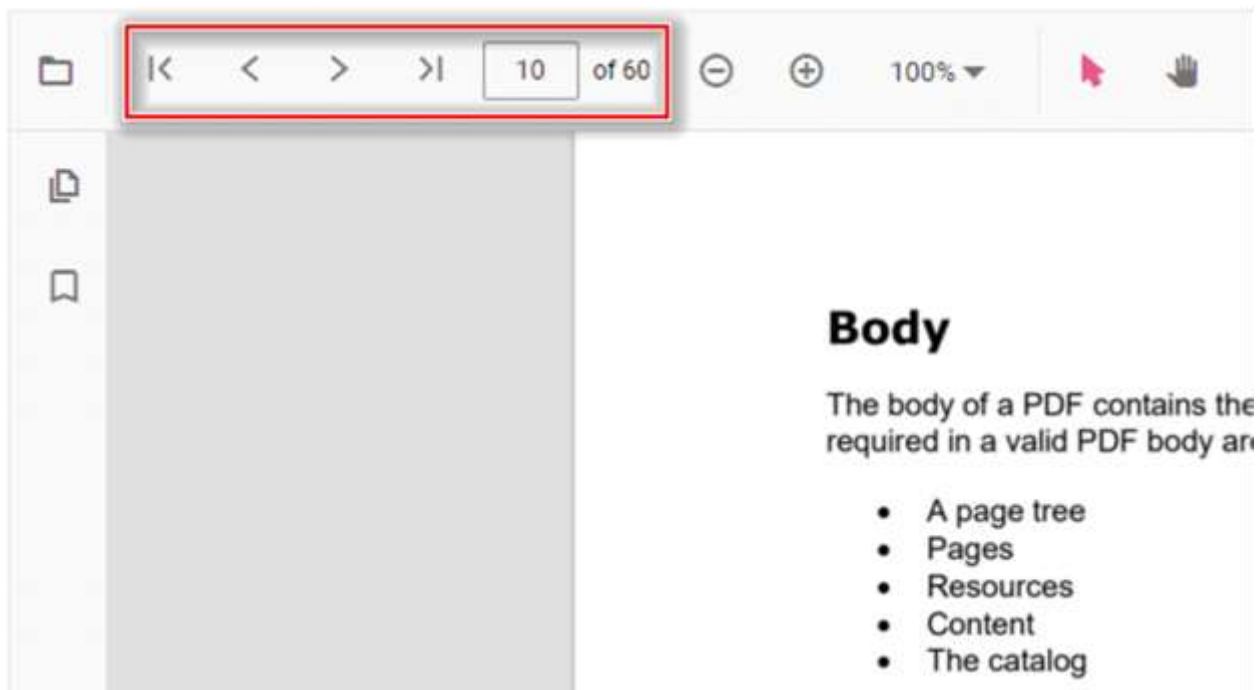
You can enable/disable page navigation option in PDF Viewer using the following code snippet.

STANDALONE

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableNavigation(true).DocumentPath("http
s://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).EnableNavigation(true).DocumentPath("https://cdn.syncfu
sion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```



Bookmark navigation

The bookmarks saved in PDF files are loaded and made ready for easy navigation.

You can enable/disable bookmark navigation by using the following code snippet.,

STANDALONE

```
```html
<div style="width:100%;height:600px">
```

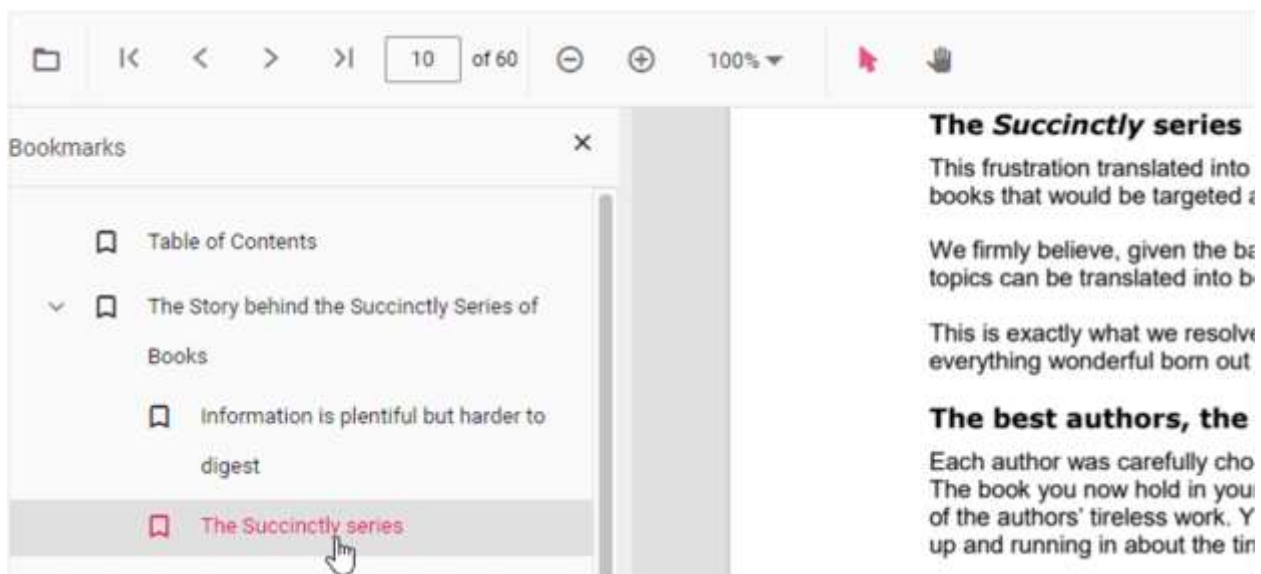
```
@Html.EJS().PdfViewer("pdfviewer").EnableBookmark(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```

```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).EnableBookmark(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```

```



Thumbnail navigation

Thumbnails is the miniature representation of actual pages in PDF files. This feature displays thumbnails of the pages and allows navigation.

You can enable/disable thumbnail navigation by using the following code snippet.,

STANDALONE

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableThumbnail(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```

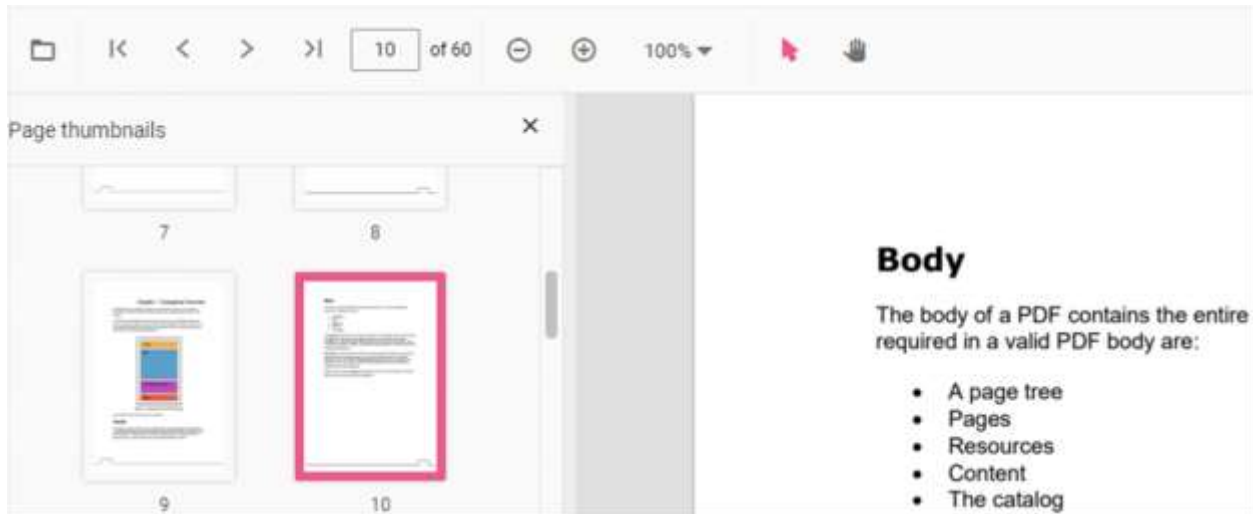
```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">

```

```
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
 "~/api/PdfViewer/") .EnableThumbnail(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
....
```



### Hyperlink navigation

Hyperlink navigation features enables navigation to the URLs (website links) in a PDF file.



### Table of content navigation

Table of contents navigation allows users to navigate to different parts of a PDF file that are listed in the table of contents section.

You can enable/disable link navigation by using the following code snippet.,

#### **STANDALONE**

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableHyperlink(true).DocumentPath("https
://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```



```

<\/div>

```

SERVER-BACKED

```

<\/div>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/") ).EnableHyperlink(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
<\/div>

```

You can change the open state of the hyperlink in the PDF Viewer by using the following code snippet.,

STANDALONE

```

<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").HyperlinkOpenState(Syncfusion.EJ2.PdfViewer.LinkTarget.NewTab).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
<\/div>

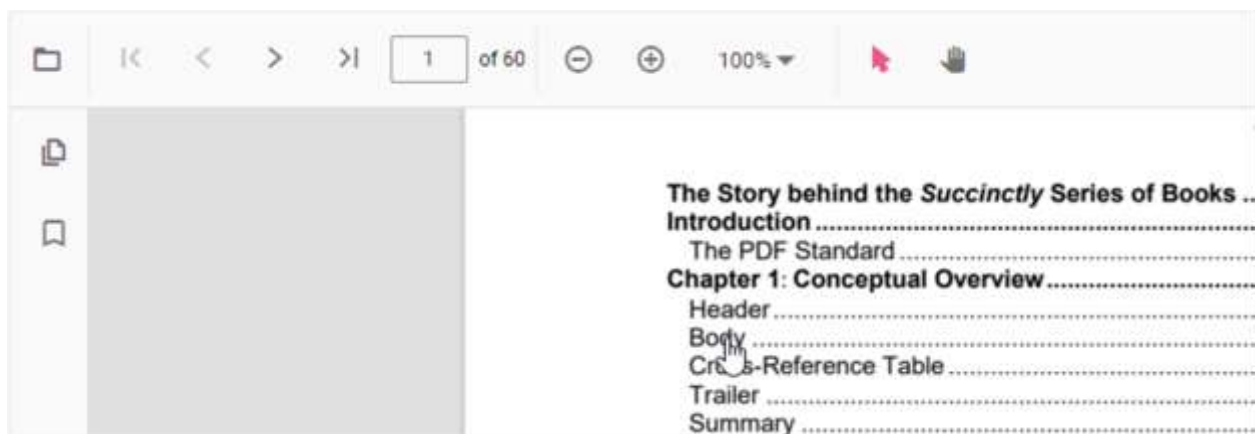
```

SERVER-BACKED

```

<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/") ).HyperlinkOpenState(Syncfusion.EJ2.PdfViewer.LinkTarget.NewTab).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
<\/div>

```



See also

- [Toolbar items](#)
- [Feature Modules](#)

Magnification

The magnification tools of the PDF viewer contains ZoomIn,ZoomOut,Zoom,FitPage, and FitWidth tools in the default toolbar. The PDF Viewer also has an option to show or hide the magnification tools in the default toolbar.

The following code snippet describes how to enable the magnification in PDF Viewer.

STANDALONE

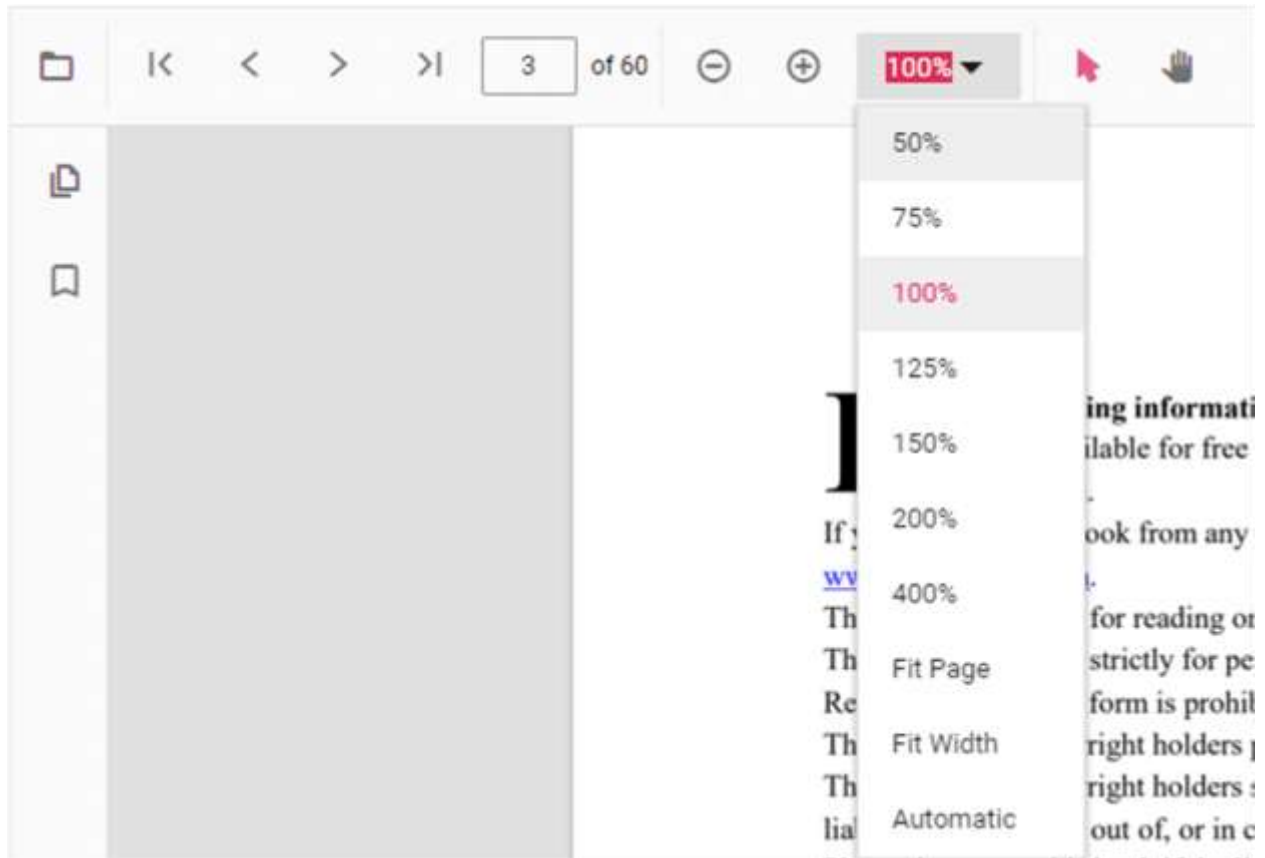
```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableMagnification(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/")).EnableMagnification(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

The following magnification options are available in the default toolbar of PDF Viewer:-

- **ZoomIn**:- perform Zoom in from current zoom value of PDF pages.
- **ZoomOut**:- perform Zoom out from current zoom value of PDF pages.
- **Zoom**:- Zoom to specific zoom value of PDF pages.
- **FitPage**:- Fits the page width with-in the available view port size.
- **FitWidth**:- Fits the view port width based on the page content size.
- **Auto**:- Fits the page content with-in the viewport on resizing action.



Note: PDF Viewer can support the zoom value ranges from 50 to 400.

See also

- [Toolbar items](#)
- [Feature Modules](#)

Text Search

The Text Search option in PDF Viewer is used to find and highlight the text content from the document. You can enable/disable the text search using the following code snippet.

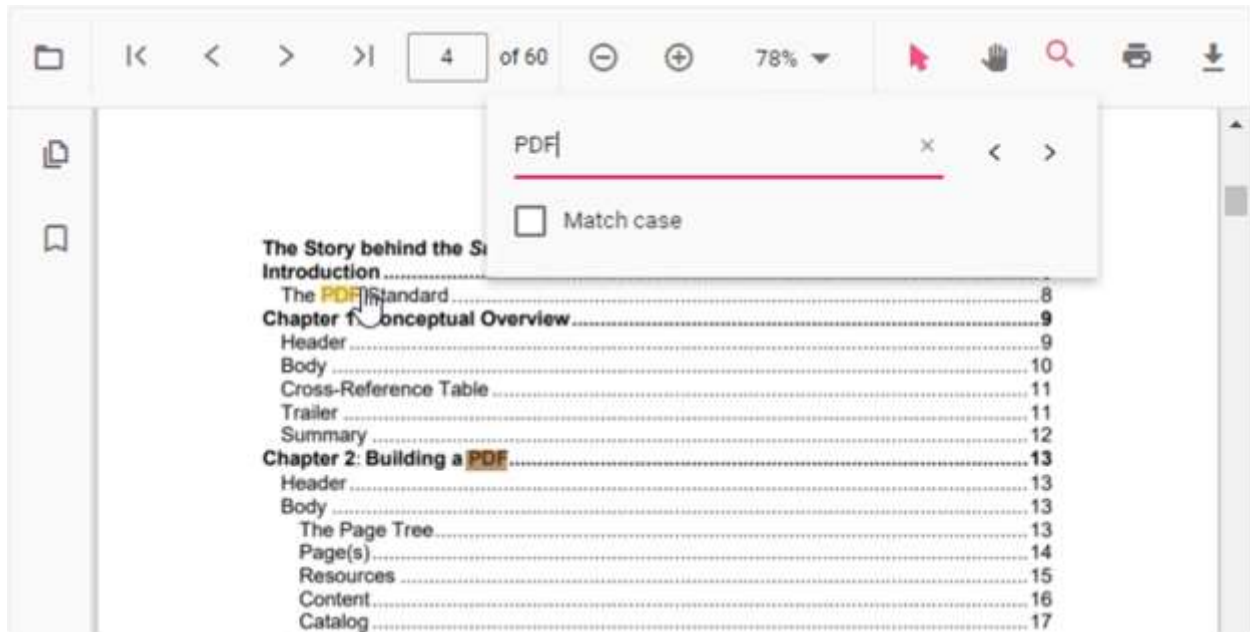
STANDALONE

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableTextSearch(true).DocumentPath("http
s://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
```

```
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
 "~/api/PdfViewer/")).EnableTextSearch(true).DocumentPath("https://cdn.syncfu
 sion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
....
```



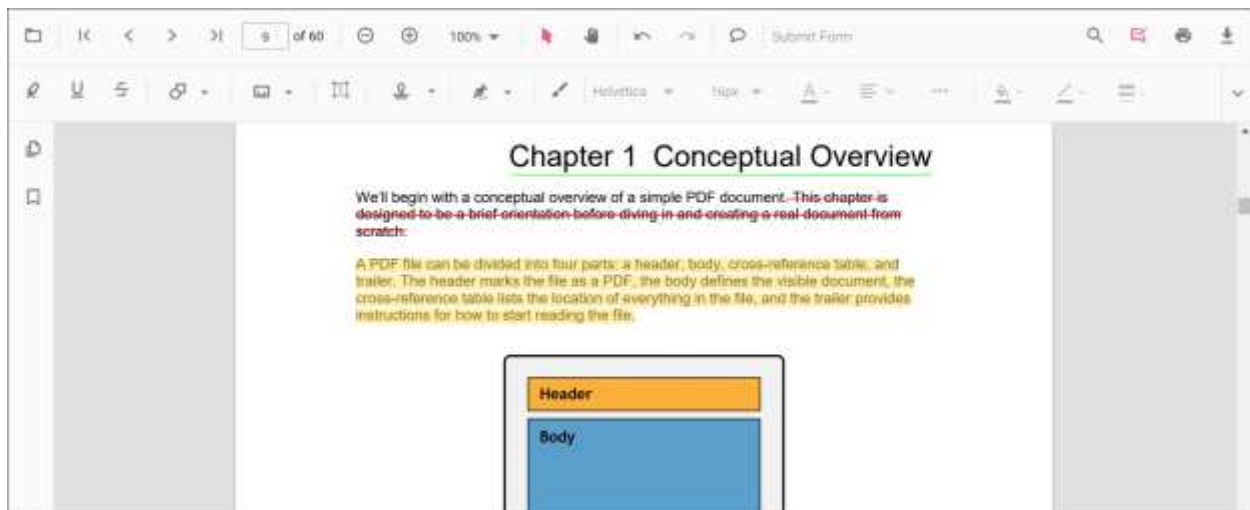
See also

- [Toolbar items](#)
- [Feature Modules](#)

## Annotation

Text Markup Annotation in the ASP.NET MVC PDF Viewer component

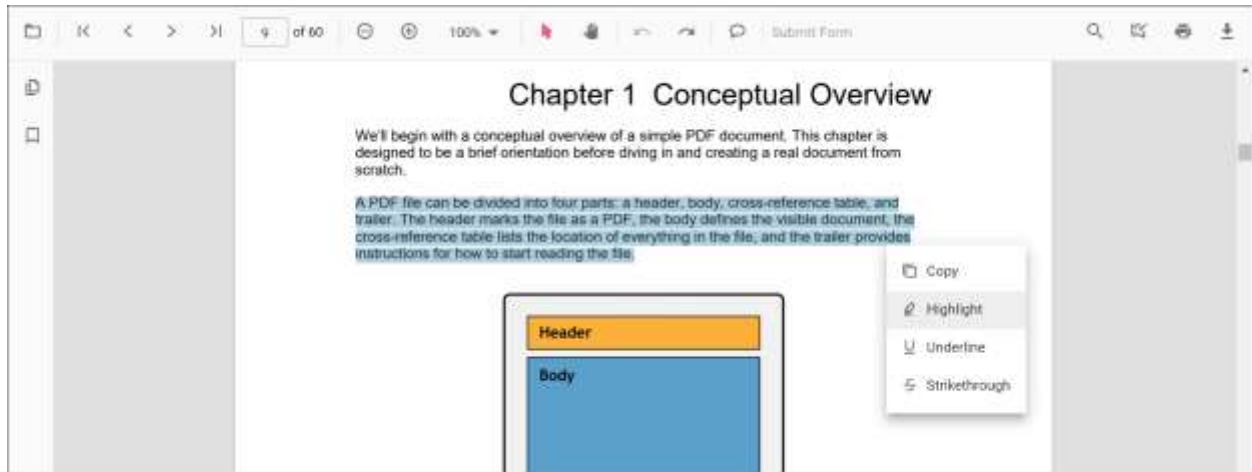
The PDF Viewer control provides the options to add, edit, and delete text markup annotations such as highlight, underline, and strikethrough annotations in the PDF document.



*Highlight a text*

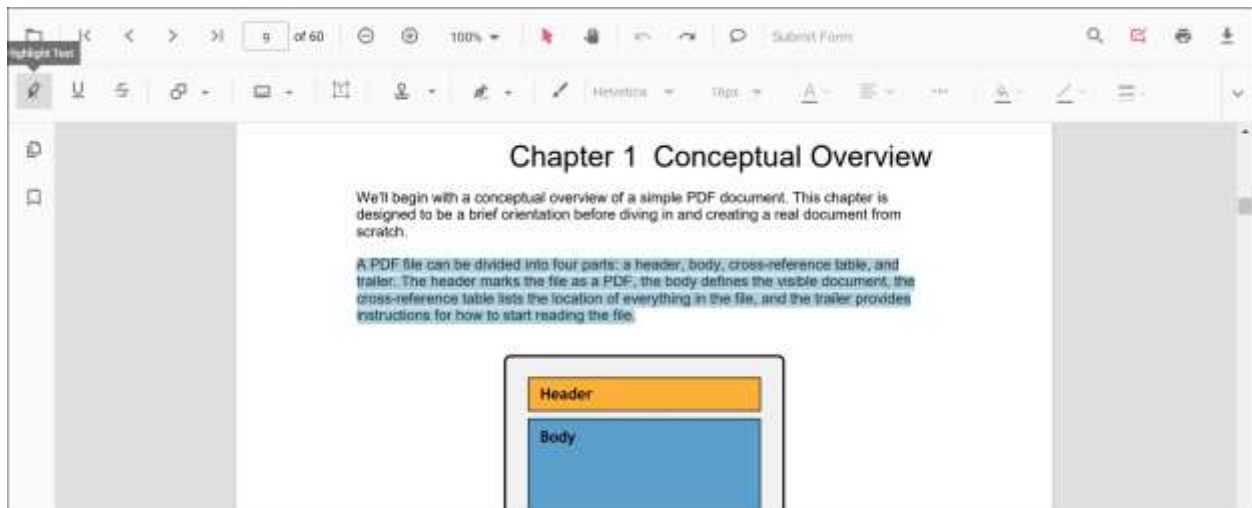
There are two ways to highlight a text in the PDF document:

1. Using the context menu
  - Select a text in the PDF document and right-click it.
  - Select **Highlight** option in the context menu that appears.



<!-- markdownlint-disable MD029 -->

2. Using the annotation toolbar
  - Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
  - Select the **Highlight** button in the annotation toolbar. It enables the highlight mode.
  - Select the text and the highlight annotation will be added.
  - You can also select the text and apply the highlight annotation using the **Highlight** button.



In the pan mode, if the highlight mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for highlighting the text.

Refer to the following code sample to switch to the highlight mode.

**STANDALONE**

```

```html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Highlight</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Highlight');
}
</script>
```

```

**SERVER-BACKED**

```

```html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Highlight</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Highlight');
}
</script>
```

```

Refer to the following code sample to switch back to normal mode from the highlight mode.

**STANDALONE**

```

```html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Highlight</button>
<!--Element to set normal mode-->
<button id="setNone" onclick="setNone()">Normal Mode</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Highlight');
}
function setNone() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('None');
}

```

```
}
</script>
...

```

SERVER-BACKED

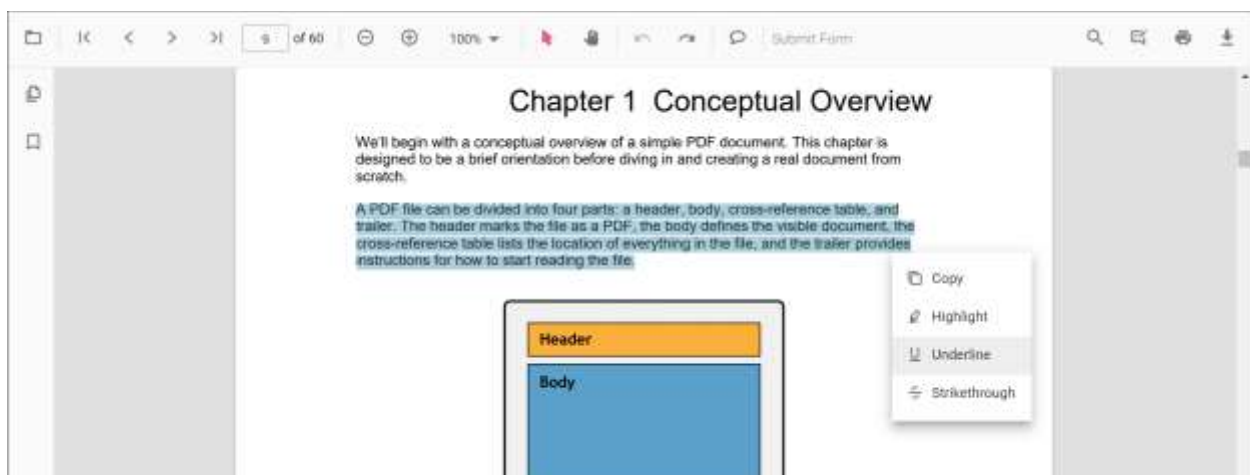
```
```html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Highlight</button>
<!--Element to set normal mode-->
<button id="setNone" onclick="setNone()">Normal Mode</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Highlight');
}
function setNone() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('None');
}
</script>
...

```

### Underline a text

There are two ways to underline a text in the PDF document:

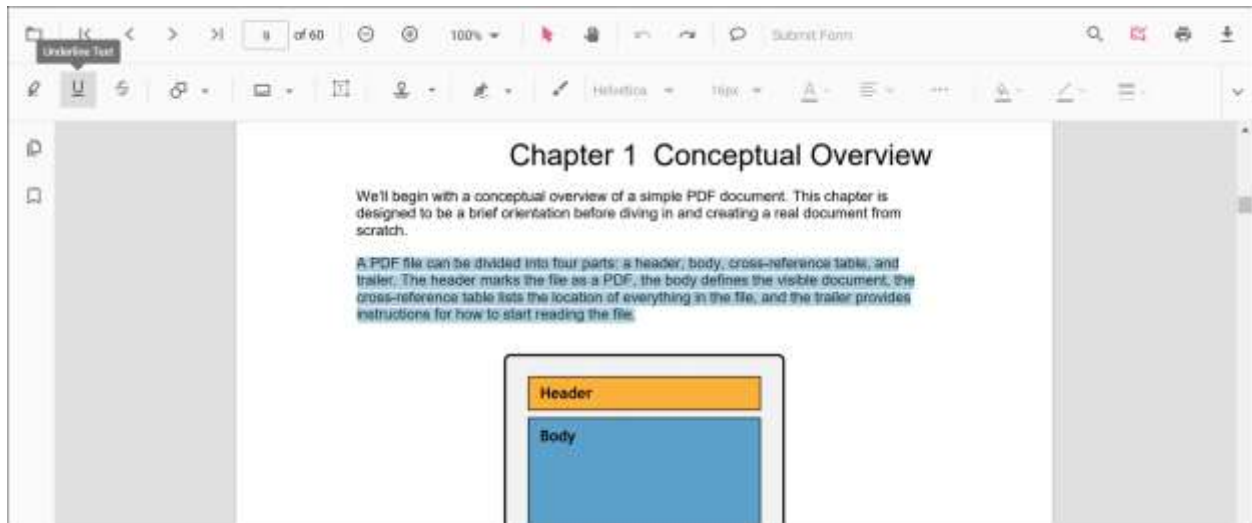
1. Using the context menu
  - o Select a text in the PDF document and right-click it.
  - o Select the **Underline** option in the context menu that appears.



```
<!-- markdownlint-disable MD029 -->
```

## 2. Using the annotation toolbar

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Underline** button in the annotation toolbar. It enables the underline mode.
- Select the text and the underline annotation will be added.
- You can also select the text and apply the underline annotation using the **Underline** button.



In the pan mode, if the underline mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for underlining the text.

Refer to the following code sample to switch to the underline mode.

### STANDALONE

```

```html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Underline</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Underline');
}
</script>
```

```

### SERVER-BACKED

```

```html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Underline</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()

```



```

</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Underline');
}
</script>
````

```

Refer to the following code sample to switch back to normal mode from the underline mode.

### **STANDALONE**

```

````html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Underline</button>
<!--Element to set normal mode-->
<button id="setNone" onclick="setNone()">Normal Mode</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Underline');
}
function setNone() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('None');
}
</script>
````

```

### **SERVER-BACKED**

```

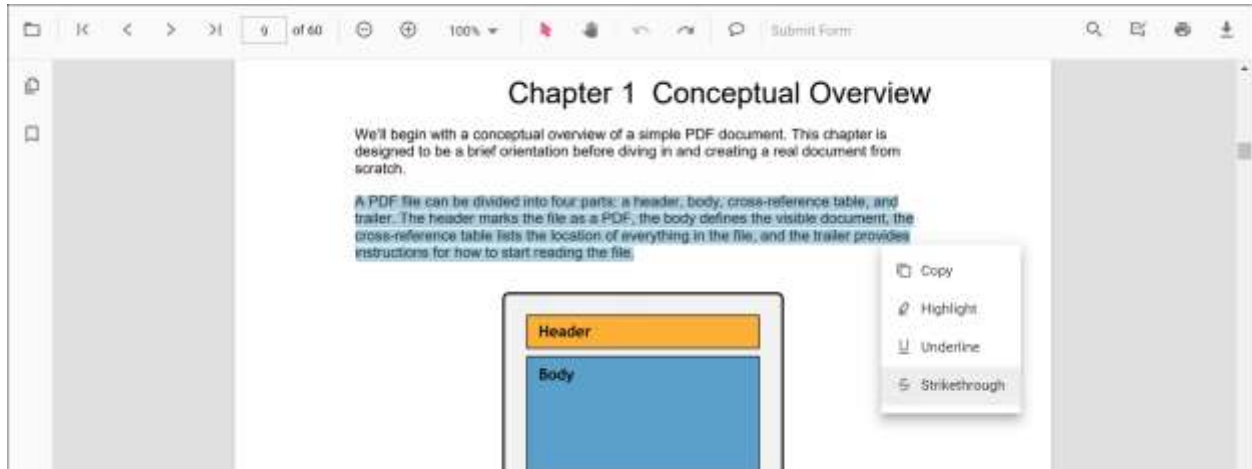
````html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Underline</button>
<!--Element to set normal mode-->
<button id="setNone" onclick="setNone()">Normal Mode</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Underline');
}
function setNone() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('None');
}
</script>

```

Strikethrough a text

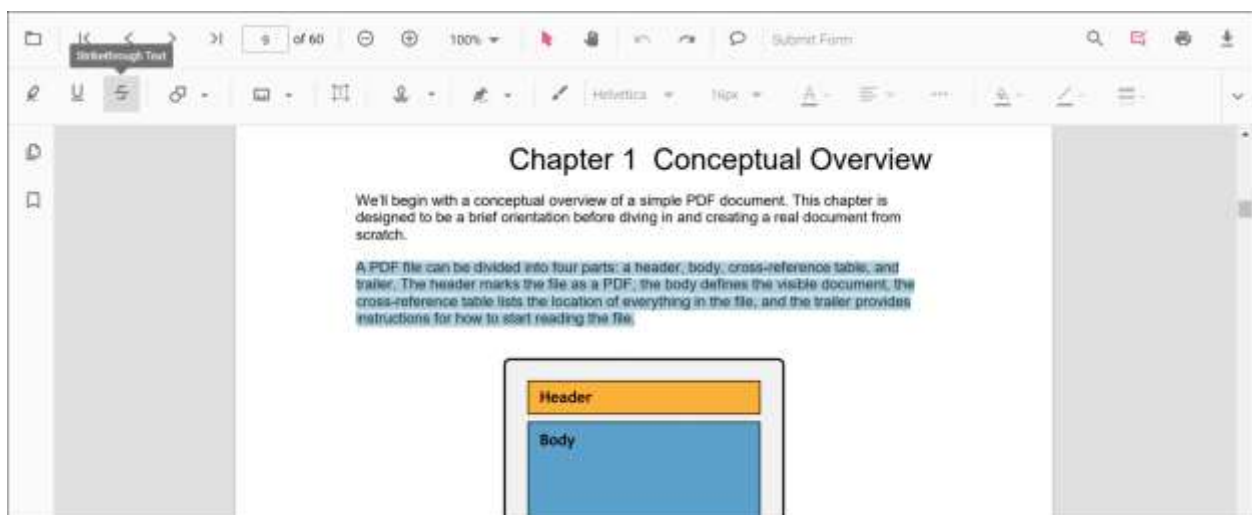
There are two ways to strikethrough a text in the PDF document:

1. Using the context menu
 - Select a text in the PDF document and right-click it.
 - Select the **Strikethrough** option in the context menu that appears.



<!-- markdownlint-disable MD029 -->

2. Using the annotation toolbar
 - Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
 - Select the **Strikethrough** button in the annotation toolbar. It enables the strikethrough mode.
 - Select the text and the strikethrough annotation will be added.
 - You can also select the text and apply the strikethrough annotation using the **Strikethrough** button.



In the pan mode, if the strikethrough mode is entered, the PDF Viewer control will switch to text select mode to enable the text selection for striking through the text.

Refer to the following code sample to switch to the strikethrough mode.

STANDALONE

```

` ``html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Strikethrough</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Strikethrough');
}
</script>
` ``

```

SERVER-BACKED

```

` ``html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Strikethrough</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Strikethrough');
}
</script>
` ``

```

Refer to the following code sample to switch back to normal mode from the strikethrough mode.

STANDALONE

```

` ``html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Strikethrough</button>
<!--Element to set normal mode-->
<button id="setNone" onclick="setNone()">Normal Mode</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Strikethrough');
}

```

```

}
function setNone() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('None');
}
</script>
` ``

```

SERVER-BACKED

```

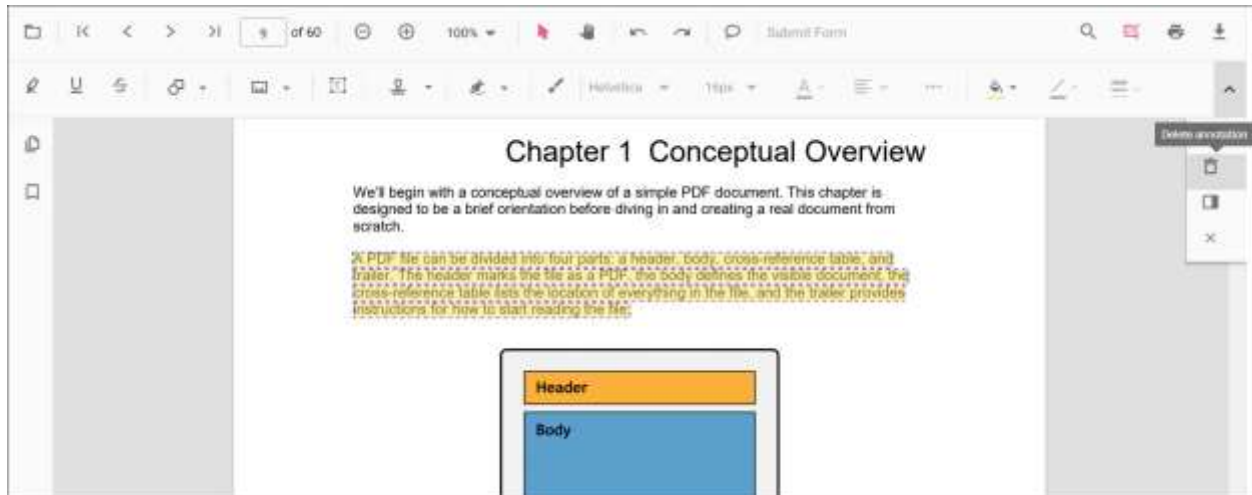
` ``html
<!--Element to set text markup annotation mode-->
<button id="set" onclick="addAnnot()">Strikethrough</button>
<!--Element to set normal mode-->
<button id="setNone" onclick="setNone()">Normal Mode</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Strikethrough');
}
function setNone() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('None');
}
</script>
` ``

```

Deleting a text markup annotation

The selected annotation can be deleted in the following ways:

1. Using the Delete key
 - Select the annotation to be deleted.
 - Click the Delete key in the keyboard. The selected annotation will be deleted.
2. Using the annotation toolbar
 - Select the annotation to be deleted.
 - Click the **Delete Annotation** button in the annotation toolbar. The selected annotation will be deleted.

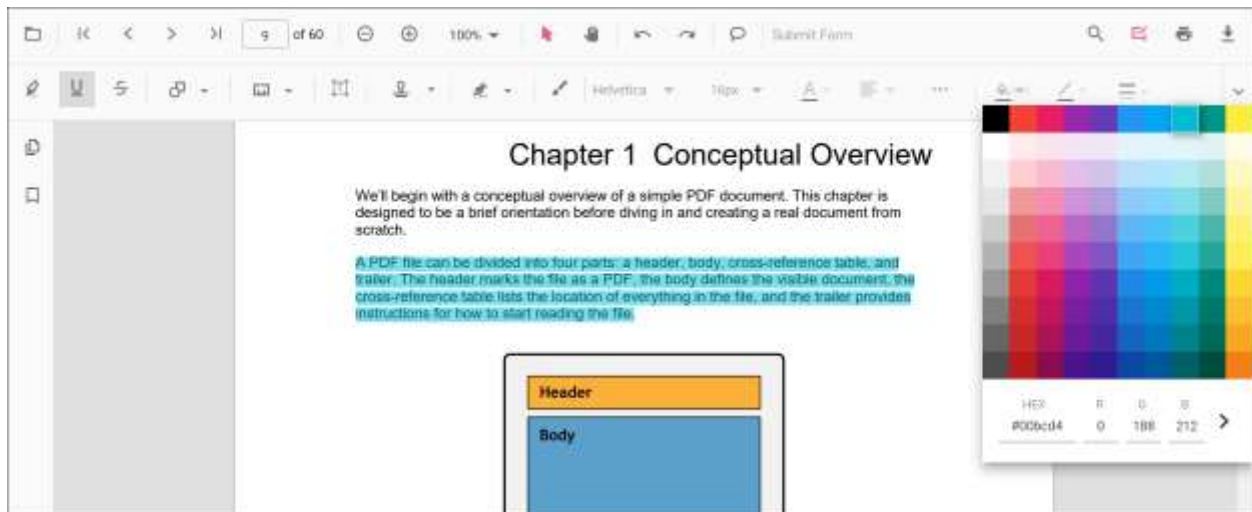


Editing the properties of the text markup annotation

The color and the opacity of the text markup annotation can be edited using the Edit Color tool and the Edit Opacity tool in the annotation toolbar.

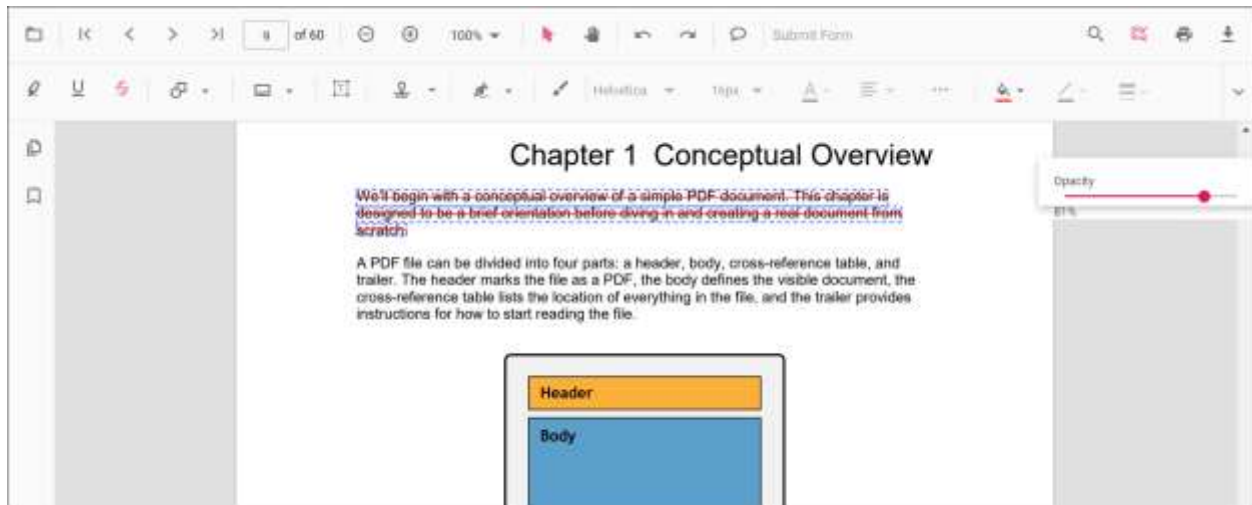
Editing color

The color of the annotation can be edited using the color palette provided in the Edit Color tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during the control initialization

The properties of the text markup annotation can be set before creating the control using the `highlightSettings`, `underlineSettings`, and `strikethroughSettings`.

Note: After editing the default color and opacity using the Edit Color tool and Edit Opacity tool, they will be changed to the selected values.

Refer to the following code sample to set the default annotation settings.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").HighlightSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerHighlightSettings{Author = "Guest User",
Color = "#ffff00", Opacity = 0.9 }).UnderlineSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerUnderlineSettings{ Author = "Guest User",
Color = "#00ffff", Opacity = 0.9 }).StrikethroughSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerStrikethroughSettings{ Author = "Guest
User", Color = "#ff00ff", Opacity = 0.9, }).Render()
</div>
<<<

```

SERVER-BACKED

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").HighlightSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerHighlightSettings{Author = "Guest User",
Color = "#ffff00", Opacity = 0.9 }).UnderlineSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerUnderlineSettings{ Author = "Guest User",
Color = "#00ffff", Opacity = 0.9 }).StrikethroughSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerStrikethroughSettings{ Author = "Guest
User", Color = "#ff00ff", Opacity = 0.9, }).Render()
</div>
<<<

```

Performing undo and redo

The PDF Viewer performs undo and redo for the changes made in the PDF document. In the text markup annotation, undo and redo actions are provided for:

- Inclusion of the text markup annotations.
- Deletion of the text markup annotations.
- Change of either color or opacity of the text markup annotations.

The undo and redo actions can be done by the following ways:

1. Using the keyboard shortcuts:

After performing a text markup annotation action, you can undo it by using the Ctrl + Z shortcut and redo by using the Ctrl + Y shortcut.

2. Using toolbar:

The undo and redo can be done using the **Undo** tool and **Redo** tool provided in the toolbar.

Refer to the following code sample for calling undo and redo actions from the client-side.

STANDALONE

```

```html
<!--Element to call undo-->
<button id="undo" onclick="Undo()">Undo</button>
<!--Element to call redo-->
<button id="redo" onclick="Redo()">Redo</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function Undo() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.undo();
}
function Redo() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.redo();
}
</script>
```

```

SERVER-BACKED

```

```html
<!--Element to call undo-->
<button id="undo" onclick="Undo()">Undo</button>
<!--Element to call redo-->
<button id="redo" onclick="Redo()">Redo</button>
<div style="width:100%;height:600px">

```

```
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function Undo() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.undo();
}
function Redo() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.redo();
}
</script>
```

```

Saving the text markup annotation

When you click the download tool in the toolbar, the text markup annotations will be saved in the PDF document. This action will not affect the original document.

Printing the text markup annotation

When the print tool is selected in the toolbar, the PDF document will be printed along with the text markup annotations added to the pages. This action will not affect the original document.

Disabling text markup annotation

The PDF Viewer control provides an option to disable the text markup annotation feature. The code sample for disabling the feature is as follows.

STANDALONE

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableTextMarkupAnnotation(false).Documen
tPath("https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
</div>
```

```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).EnableTextMarkupAnnotation(false).DocumentPath("https://cdn
.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
</div>
```

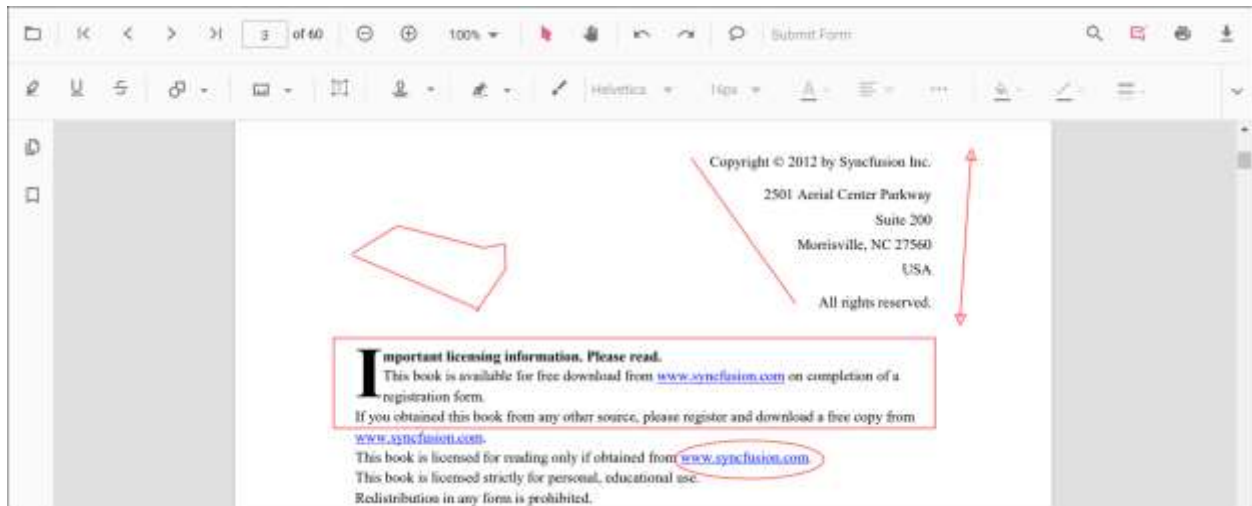
```

Shape Annotation in the ASP.NET MVC PDF Viewer component

The PDF Viewer control provides the options to add, edit, and delete the shape annotations. The shape annotation types supported in the PDF Viewer control are:

- Line
- Arrow

- Rectangle
- Circle
- Polygon

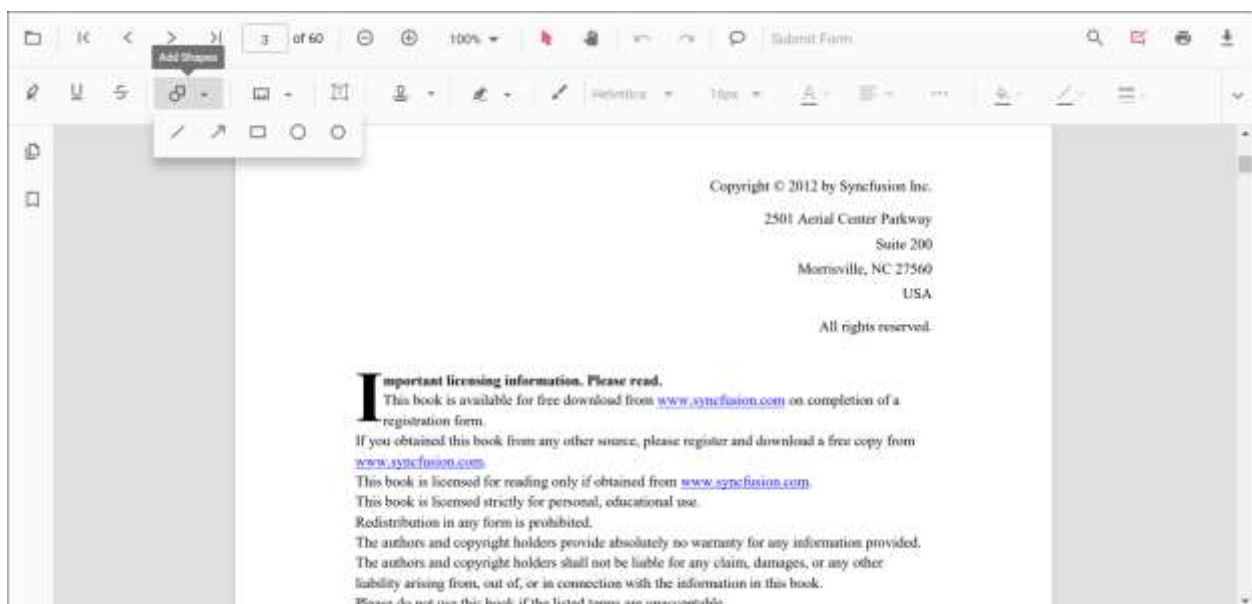


Adding a shape annotation to the PDF document

Shape annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Shape Annotation** drop-down button. A drop-down pop-up will appear and shows the shape annotations to be added.
- Select the shape types to be added to the page in the drop-down pop-up. It enables the selected shape annotation mode.
- You can add the shapes over the pages of the PDF document.

In the pan mode, if the shape annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code sample to switch to the circle annotation mode.

STANDALONE

```

` ``html
<!--Element to set shape annotation mode-->
<button id="set" onclick="addAnnot()">Circle</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Circle');
}
</script>
` ``

```

SERVER-BACKED

```

` ``html
<!--Element to set shape annotation mode-->
<button id="set" onclick="addAnnot()">Circle</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Circle');
}
</script>
` ``

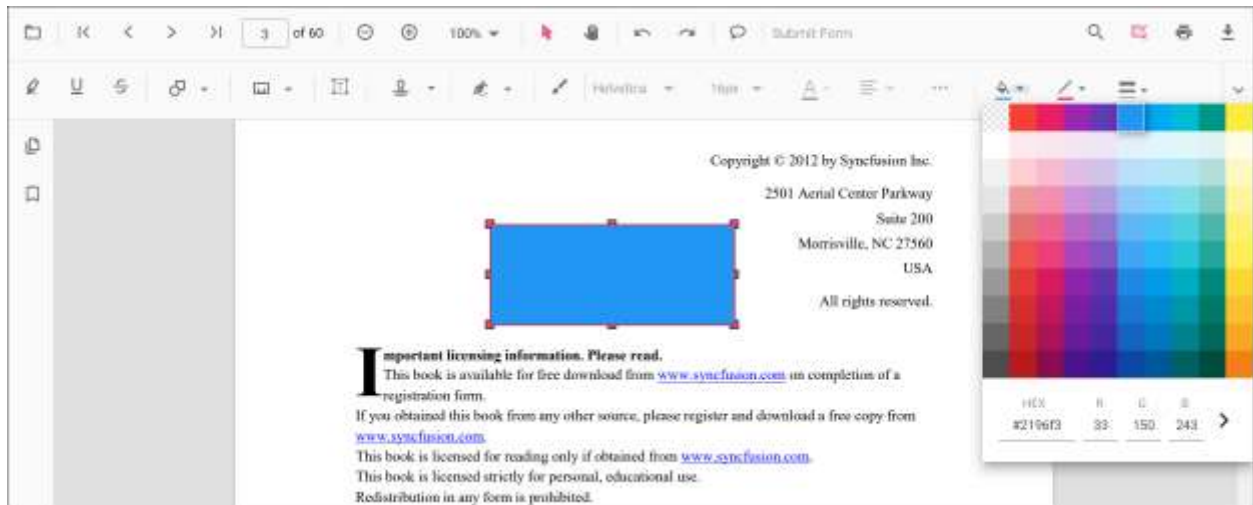
```

Editing the properties of the shape annotation

The fill color, stroke color, thickness, and opacity of the shape annotation can be edited using the Edit color tool, Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

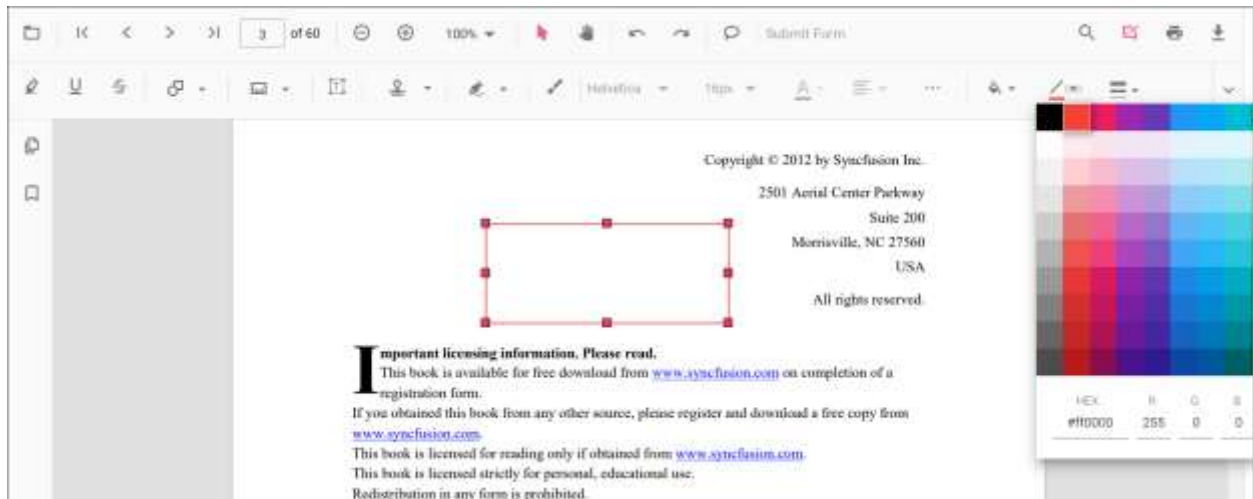
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



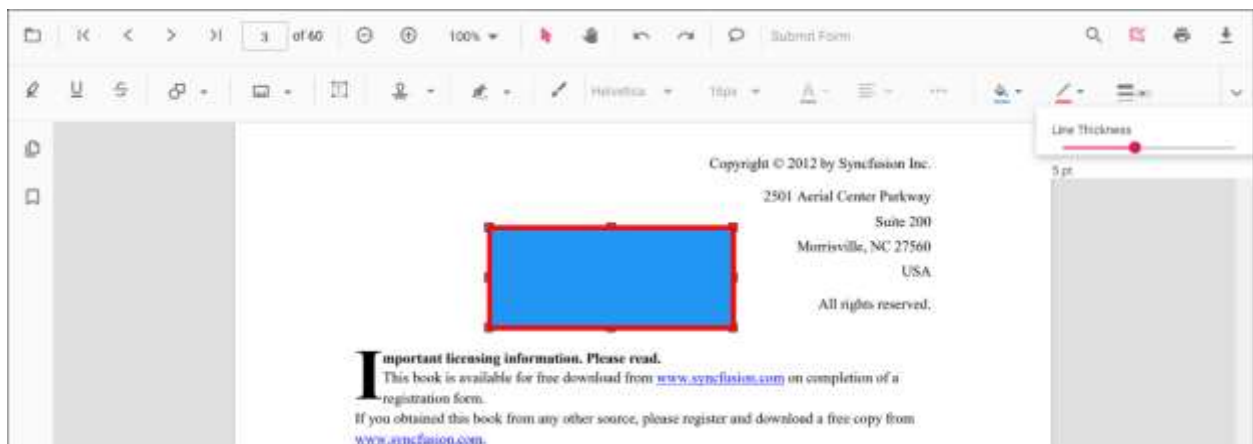
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



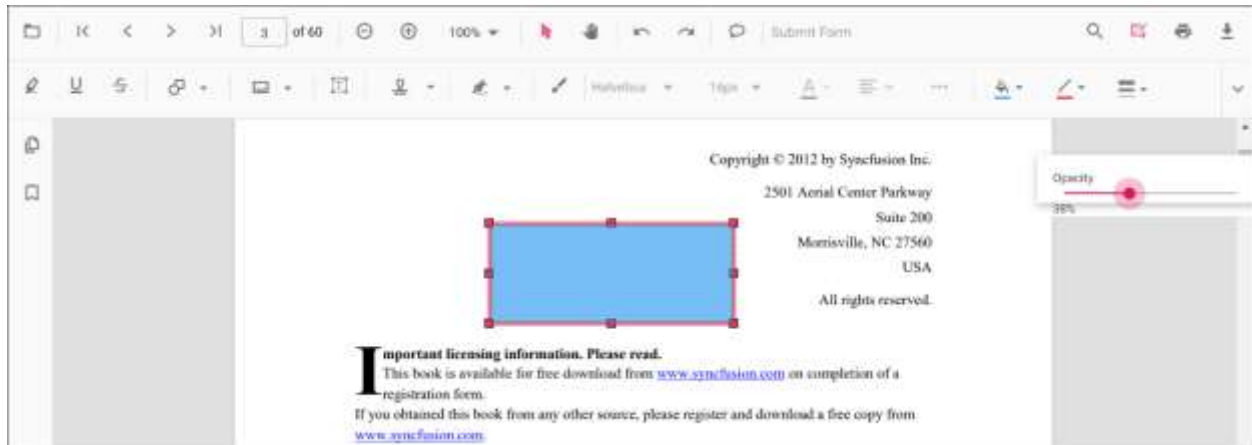
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

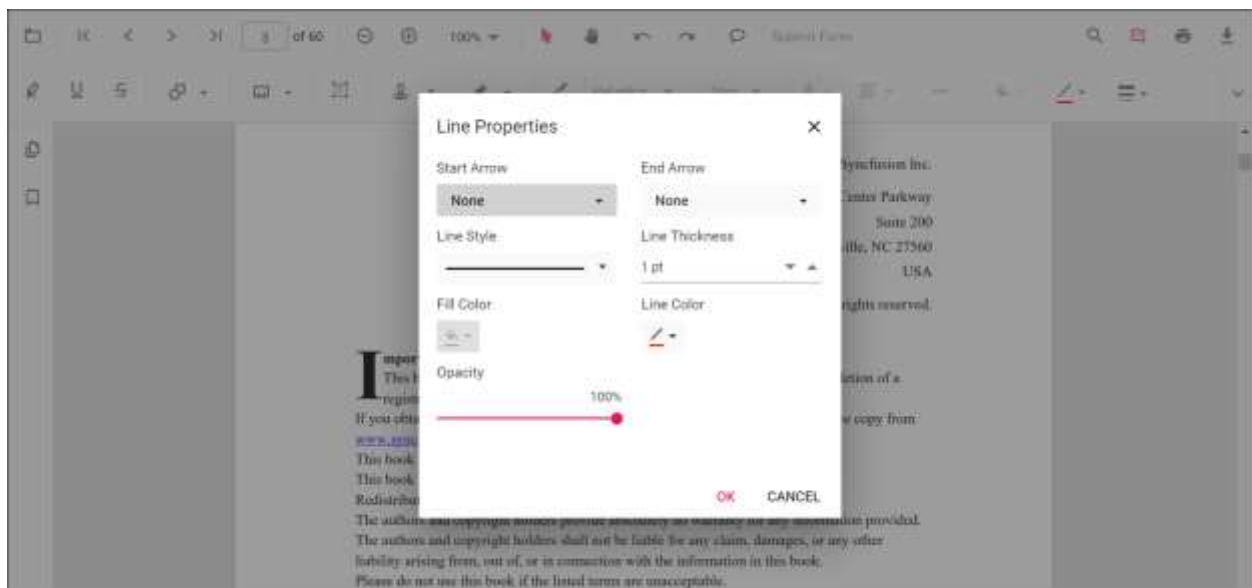
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as line and arrow annotations can be edited using the Line Properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the line and arrow annotations.

Refer to the following code sample to set the default annotation settings.



Setting default properties during the control initialization

The properties of the shape annotations can be set before creating the control using LineSettings, ArrowSettings, RectangleSettings, CircleSettings, and PolygonSettings.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").LineSettings(new
SynCFusion.EJ2.PdfViewer.PdfViewerLineSettings { FillColor = "blue", Opacity
= 0.6, StrokeColor = "green" }).ArrowSettings(new
  
```

```

Syncfusion.EJ2.PdfViewer.PdfViewerArrowSettings { FillColor = "green",
Opacity = 0.6, StrokeColor = "blue" }).RectangleSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerRectangleSettings { FillColor = "yellow",
Opacity = 0.6, StrokeColor = "orange" }).CircleSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerCircleSettings { FillColor = "orange",
Opacity = 0.6, StrokeColor = "pink" }).PolygonSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerPolygonSettings { FillColor = "pink",
Opacity = 0.6, StrokeColor = "yellow" }).Render()
</div>
` ``

```

SERVER-BACKED

```

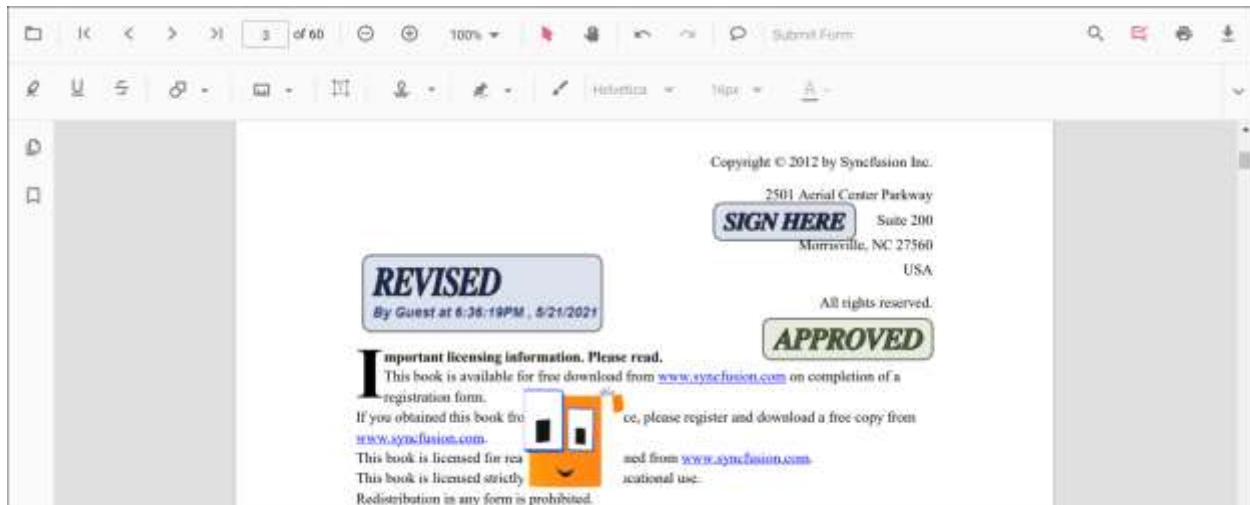
` ``html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").LineSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerLineSettings { FillColor = "blue", Opacity
= 0.6, StrokeColor = "green" }).ArrowSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerArrowSettings { FillColor = "green",
Opacity = 0.6, StrokeColor = "blue" }).RectangleSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerRectangleSettings { FillColor = "yellow",
Opacity = 0.6, StrokeColor = "orange" }).CircleSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerCircleSettings { FillColor = "orange",
Opacity = 0.6, StrokeColor = "pink" }).PolygonSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerPolygonSettings { FillColor = "pink",
Opacity = 0.6, StrokeColor = "yellow" }).Render()
</div>
` ``

```

Stamp Annotation in the ASP.NET MVC PDF Viewer component

The PDF Viewer control provides options to add, edit, delete, and rotate the following stamp annotation in the PDF documents:

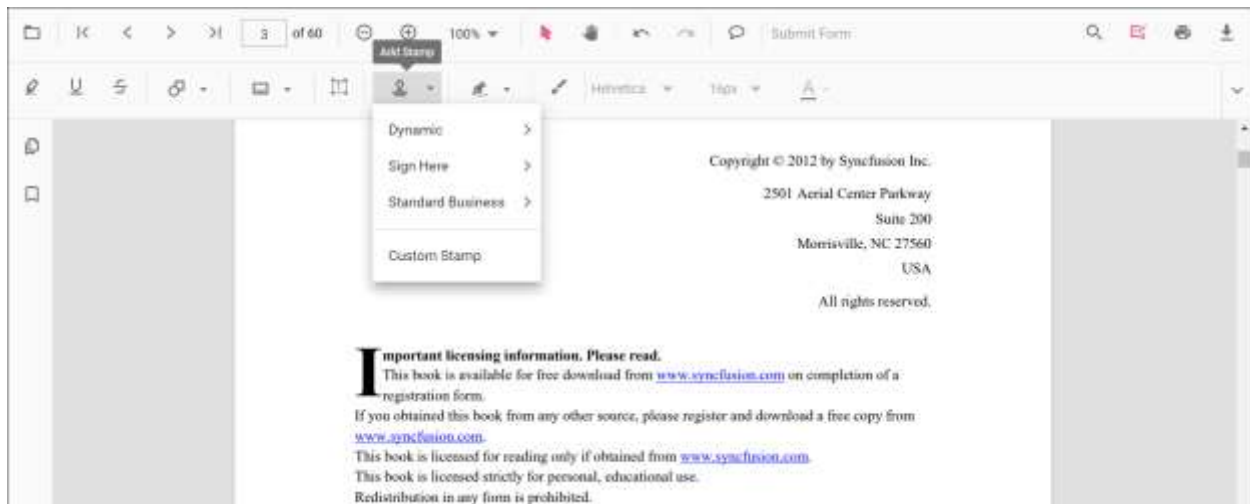
- Dynamic
- Sign Here
- Standard Business
- Custom Stamp



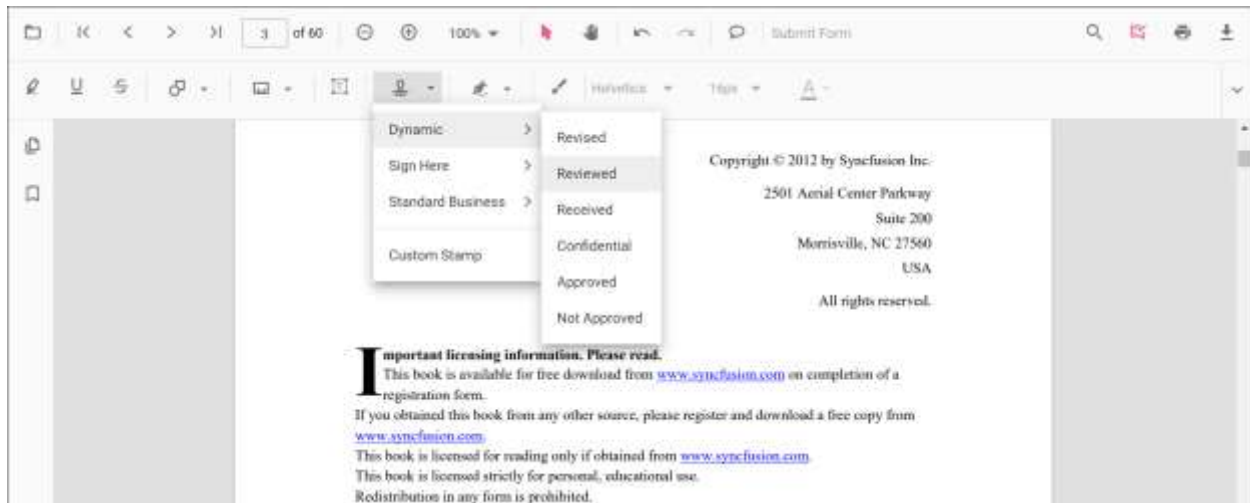
Adding stamp annotations to the PDF document

The stamp annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** drop-down button. A drop-down pop-up will appear and shows the stamp annotations to be added.



- Select the annotation type to be added to the page in the pop-up.

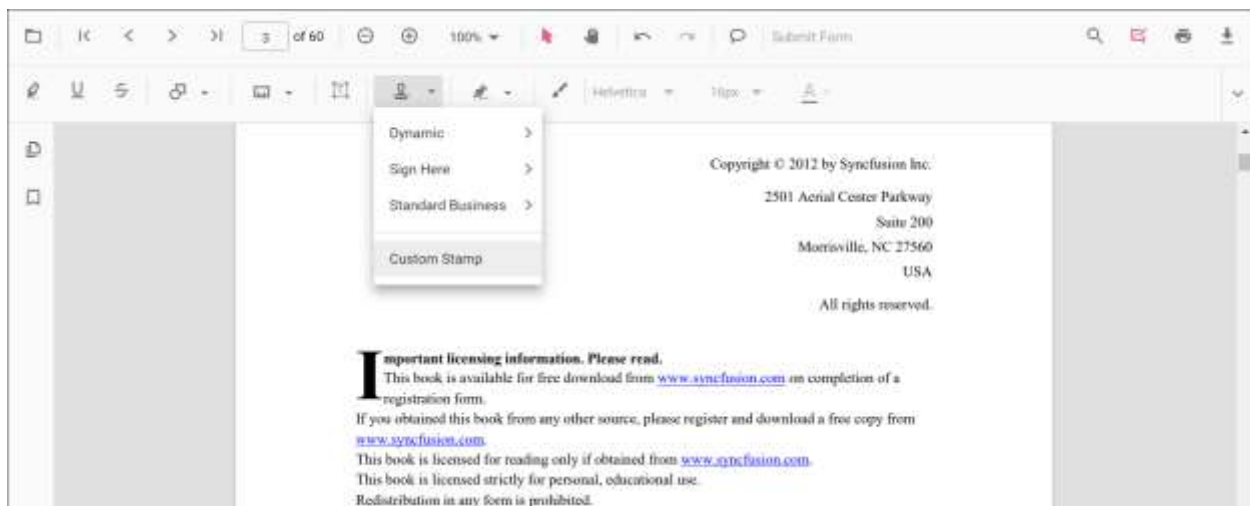


- You can add the annotation over the pages of the PDF document.

In the pan mode, if the stamp annotation mode is entered, the PDF Viewer control will switch to text select mode.

Adding custom stamp to the PDF document

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the **Stamp Annotation** drop-down button. A drop-down pop-up will appear and shows the stamp annotations to be added.
- Click the Custom Stamp button.



- The file explorer dialog will appear, choose the image and then add the image to the PDF page.

Note: The JPG and JPEG image format is only supported in the custom stamp annotations.

Setting default properties during control initialization

The properties of the stamp annotation can be set before creating the control using the StampSettings.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values. Refer to the following code sample to set the default sticky note annotation settings.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").StampSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerStampSettings { Opacity = 0.3, Author =
"Guest User" }).Render()
</div>
<<<

```

SERVER-BACKED

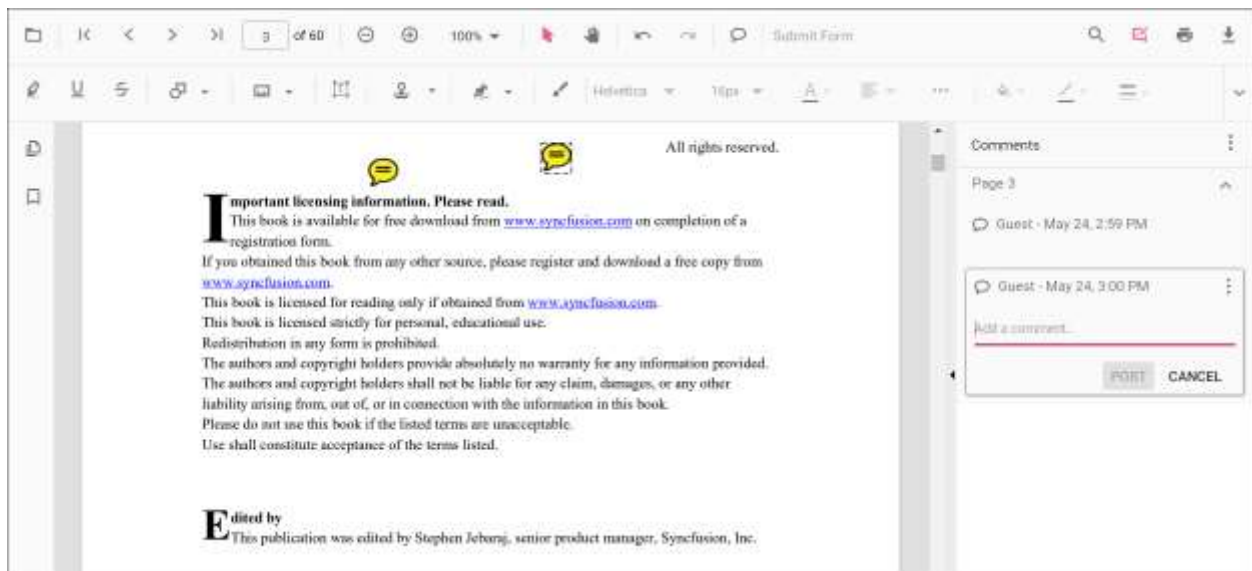
```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").StampSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerStampSettings { Opacity = 0.3, Author =
"Guest User" }).Render()
</div>
<<<

```

Sticky Notes Annotation in the ASP.NET MVC PDF Viewer component

The PDF Viewer control provides the options to add, edit, and delete the sticky note annotations in the PDF document.

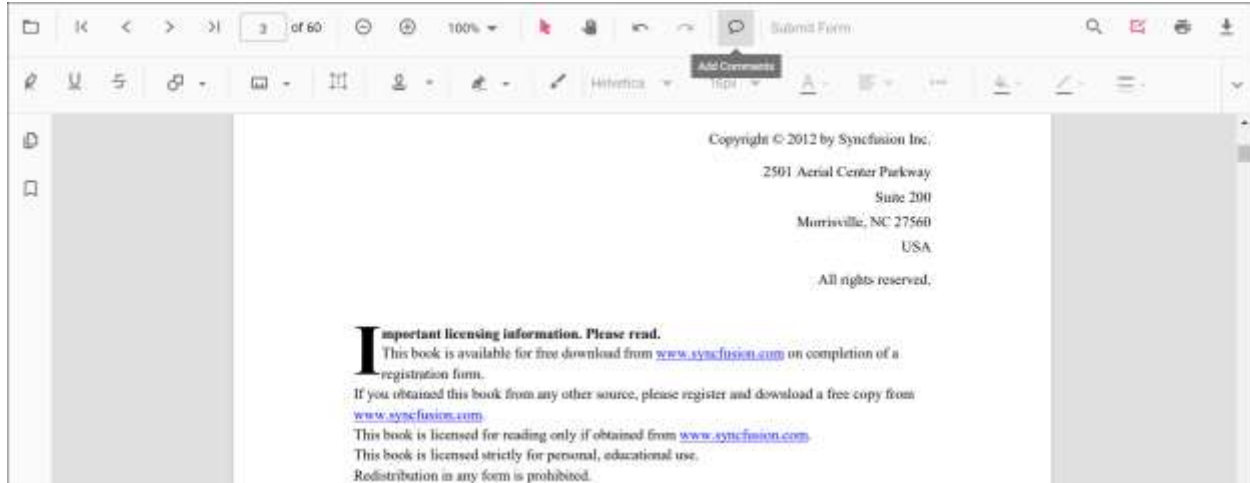


Adding a sticky note annotation to the PDF document

Sticky note annotations can be added to the PDF document using the annotation toolbar.

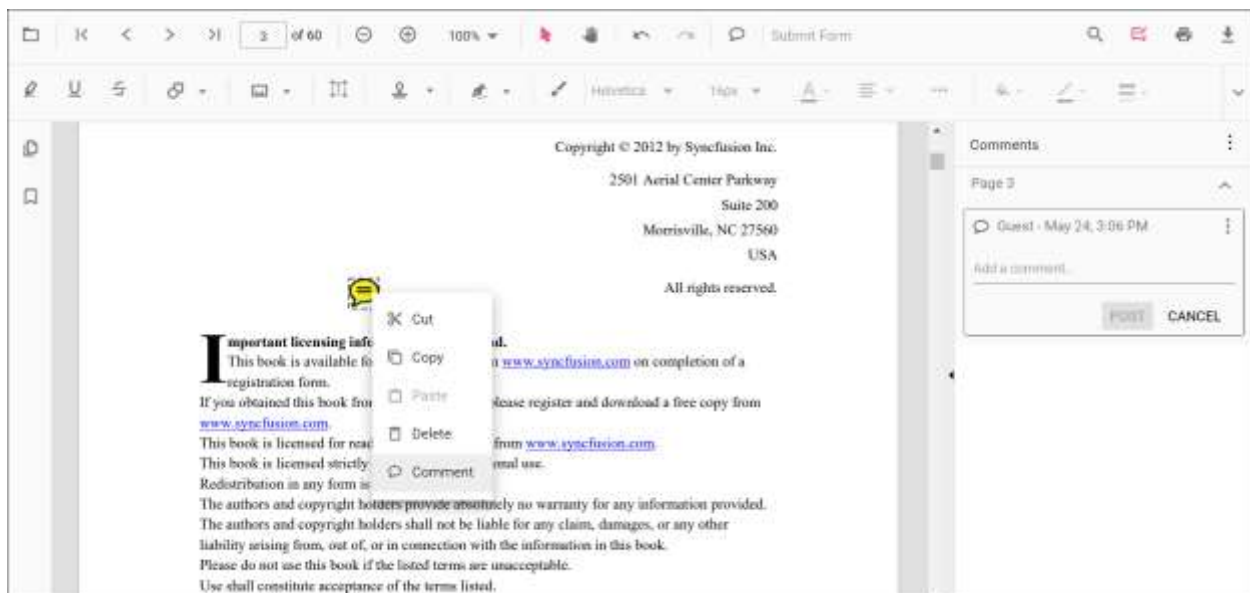
- Click the **Comments** button in the PDF Viewer toolbar. A toolbar appears below it.
- Click the position where you want to add sticky note annotation in the PDF document.

- Sticky note annotation will be added in the clicked positions.



Annotation comments can be added to the PDF document using the comment panel.

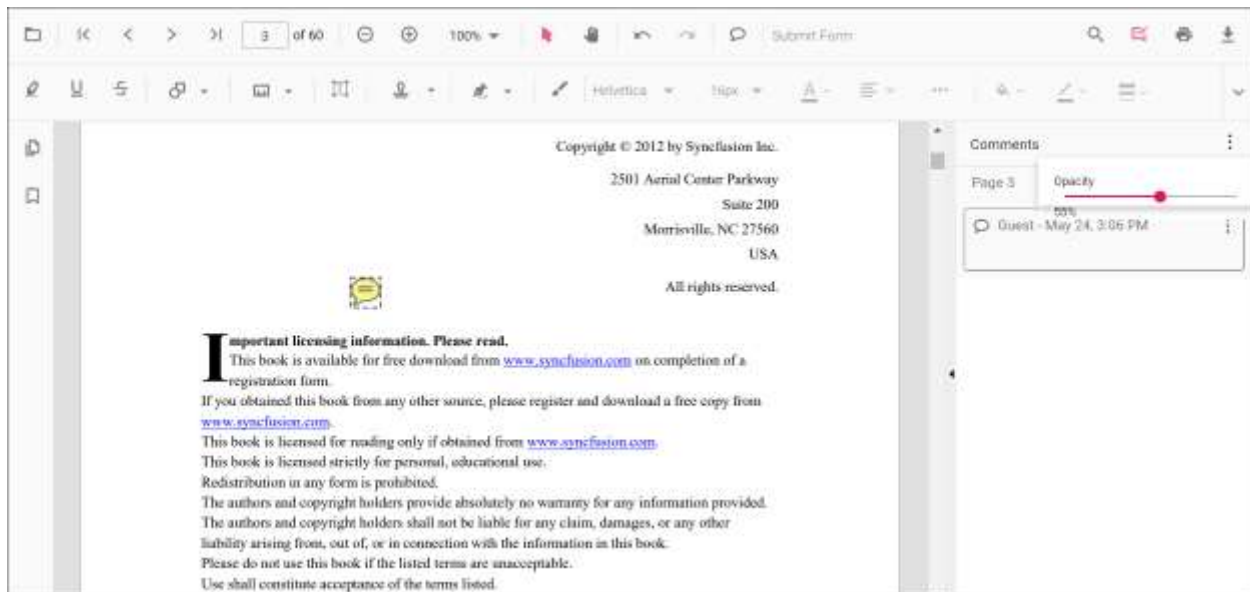
- Select a Sticky note annotation in the PDF document and right-click it.
- Select the Comment option in the context menu that appears.
- Now, you can add Comments, Reply, and Status using the Comment Panel.
- Now, you can add Comments, Reply, and Status using the Comment Panel.



Editing the properties of the sticky note annotation

Editing opacity

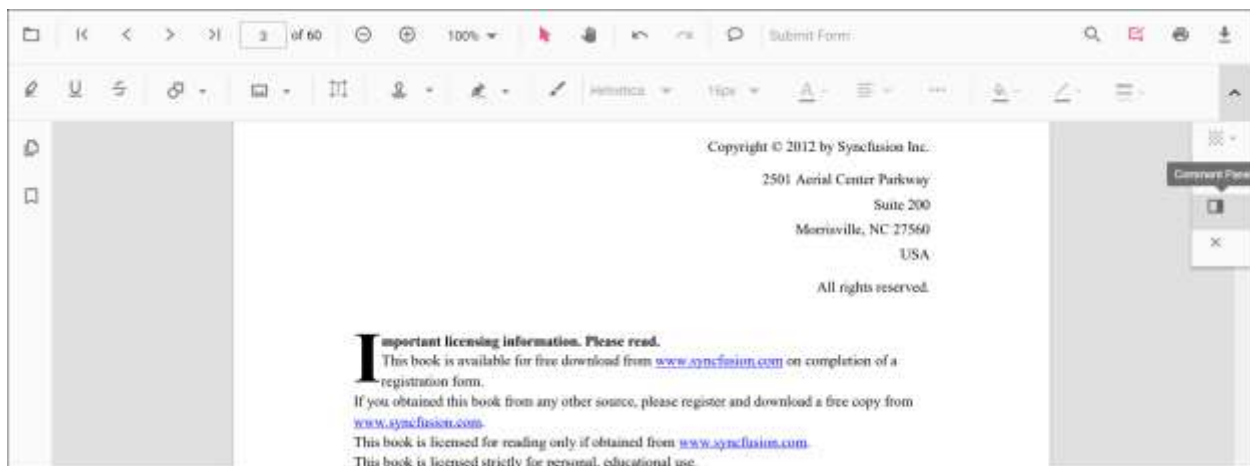
The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



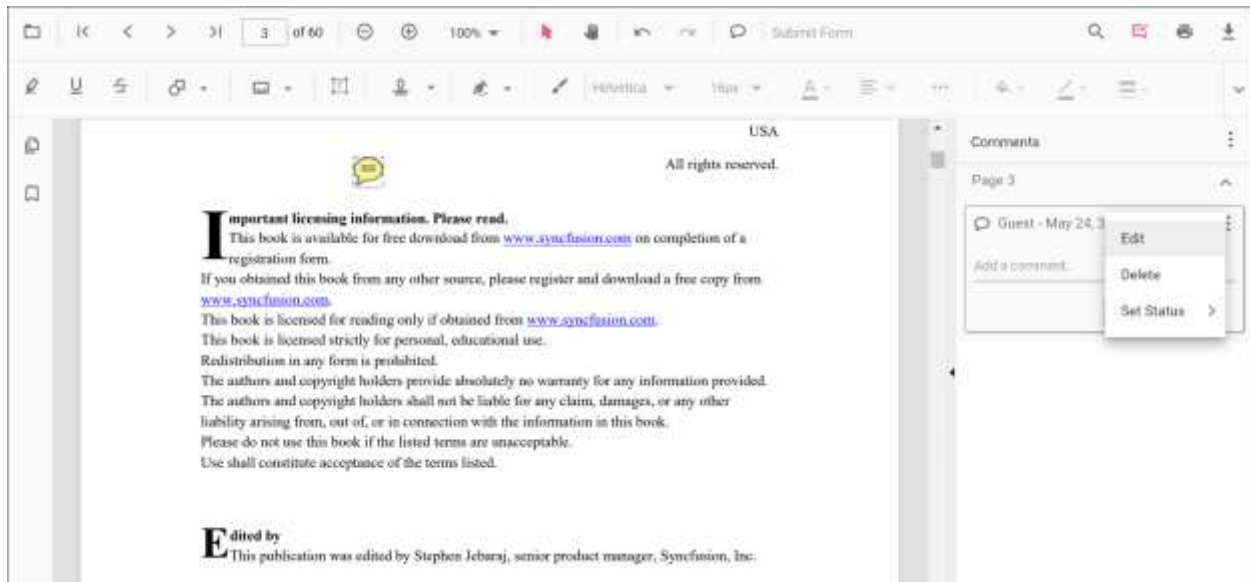
Editing comments

The comment, comment reply, and comment status of the annotation can be edited using the Comment Panel.

- Open the comment panel using the Comment Panel button showing in the annotation toolbar.



You can modify or delete the comments or comments replay and it's status using the menu option provided in the comment panel.



Setting default properties during the control initialization

The properties of the sticky note annotation can be set before creating the control using the `StickyNotesSettings`.

After editing the default opacity using the Edit Opacity tool, they will be changed to the selected values. Refer to the following code sample to set the default sticky note annotation settings.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").StickyNotesSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerStickyNotesSettings { Author =
"Syncfusion" }).Render()
</div>
<<<

```

SERVER-BACKED

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").StickyNotesSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerStickyNotesSettings { Author =
"Syncfusion" }).Render()
</div>
<<<

```

Disabling sticky note annotations

The PDF Viewer control provides an option to disable the sticky note annotations feature. The code sample for disabling the feature is as follows.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableStickyNotesAnnotation(false).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
</div>
<<<

```

SERVER-BACKED

```

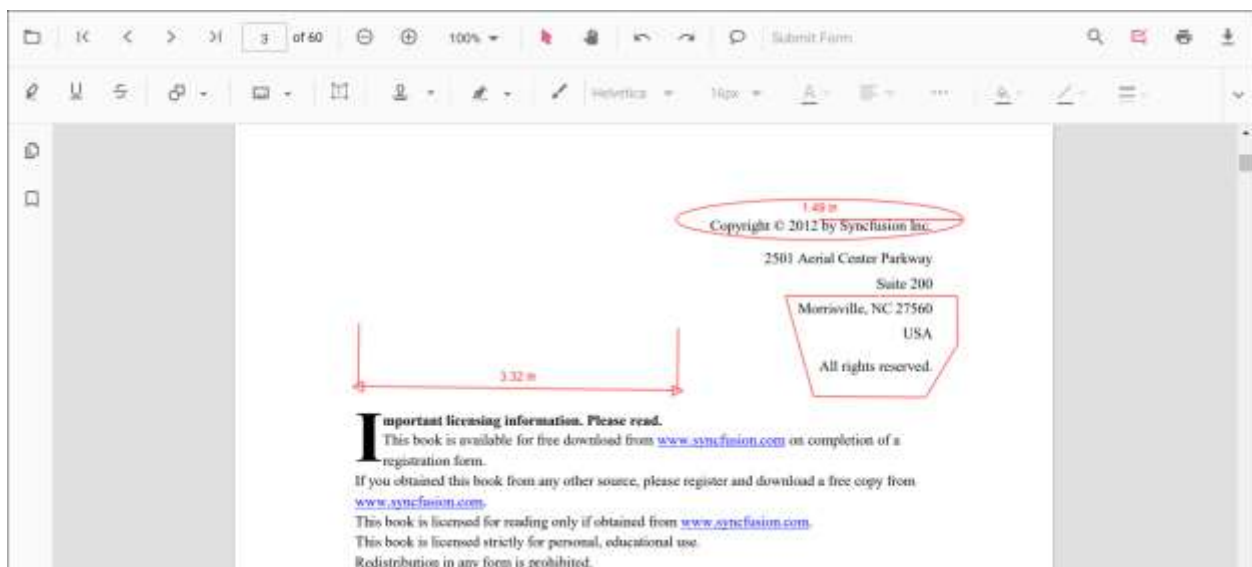
<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).EnableStickyNotesAnnotation(false).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
</div>
<<<

```

Measurement Annotation in the ASP.NET MVC PDF Viewer component

The PDF Viewer provides the options to add measurement annotations. You can measure the page annotations with the help of measurement annotation. The supported measurement annotations in the PDF Viewer control are:

- Distance
- Perimeter
- Area
- Radius
- Volume



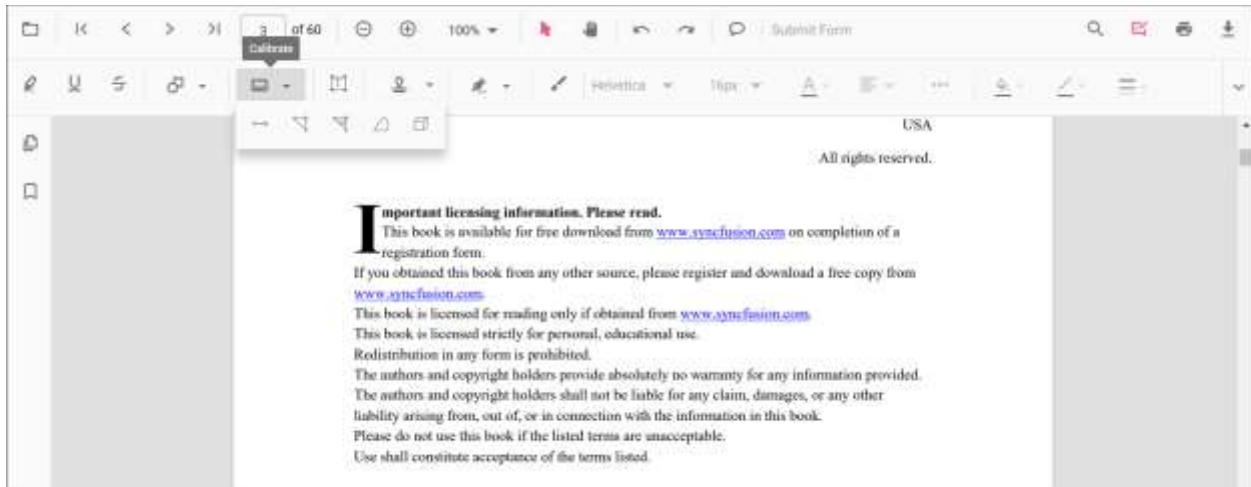
Adding a measurement annotation to the PDF document

The measurement annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.

- Click the **Measurement Annotation** dropdown button. A dropdown pop-up will appear and shows the measurement annotations to be added.
- Select the measurement type to be added to the page in the dropdown pop-up. It enables the selected measurement annotation mode.
- You can measure and add the annotation over the pages of the PDF document.

In the pan mode, if the measurement annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code sample to switch to the distance annotation mode.

STANDALONE

```

<html>
<!--Element to set measurement annotation mode-->
<button id="set" onclick="addAnnot()">Distance</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Distance');
}
</script>
</html>

```

SERVER-BACKED

```

<html>
<!--Element to set measurement annotation mode-->
<button id="set" onclick="addAnnot()">Distance</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>

```

```
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Distance');
}
</script>
...

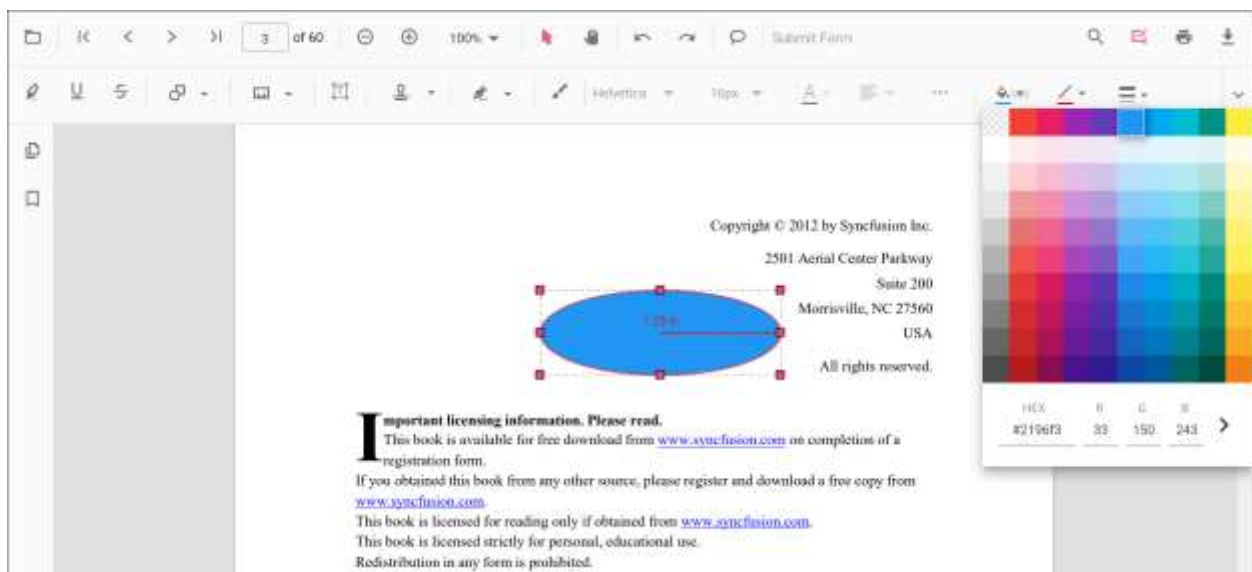
```

Editing the properties of measurement annotation

The fill color, stroke color, thickness, and opacity of the measurement annotation can be edited using the Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

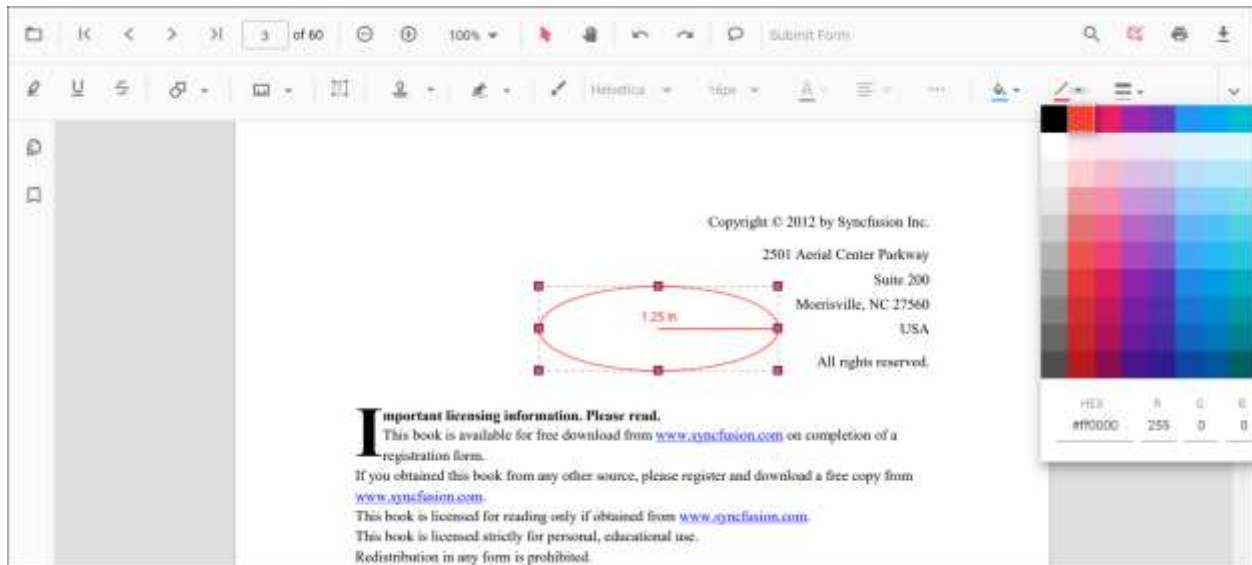
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



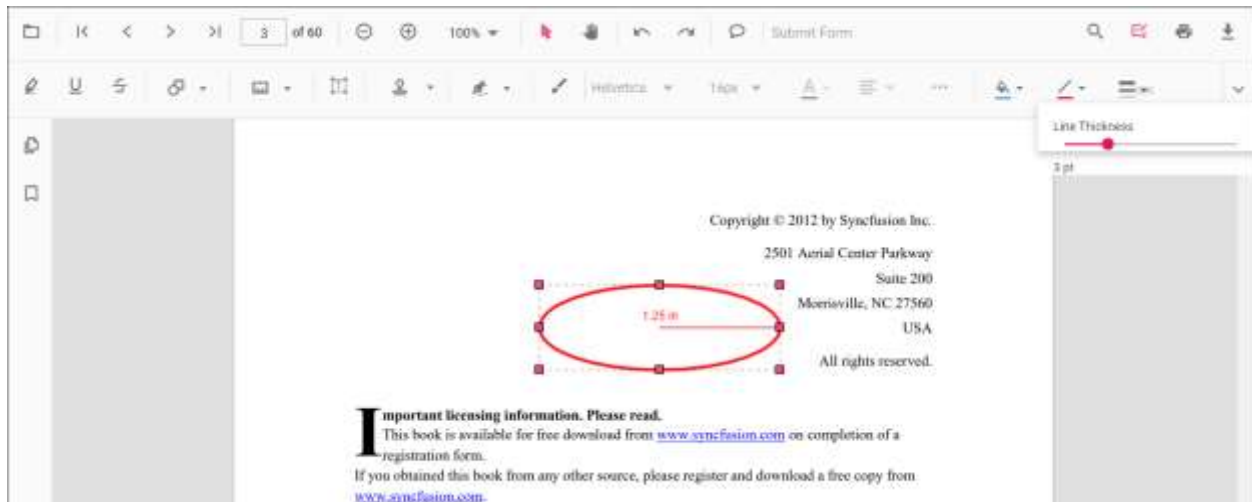
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



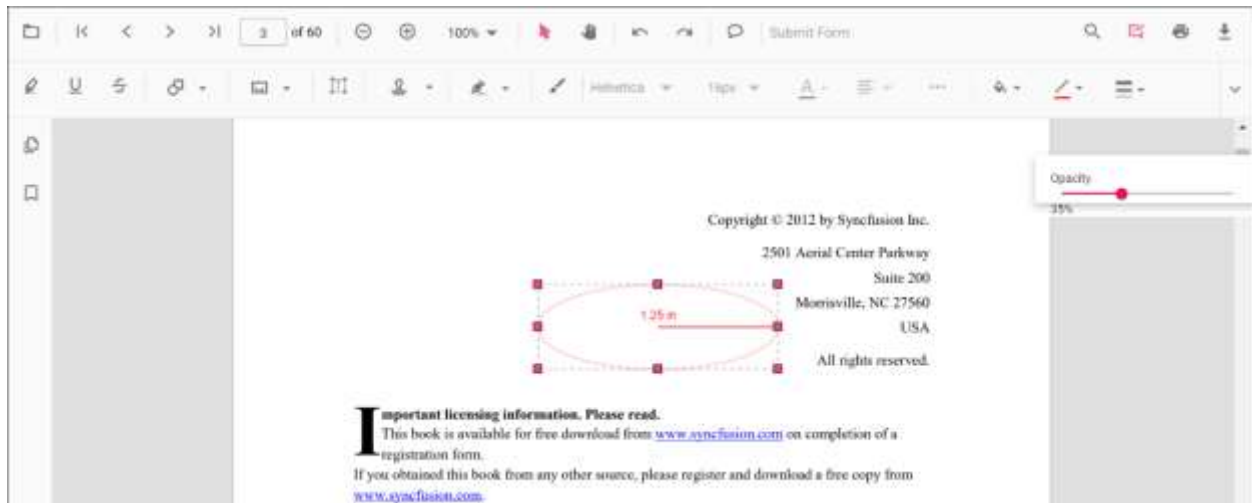
Editing thickness

The thickness of the border of the annotation can be edited using the range slider provided in the Edit thickness tool.



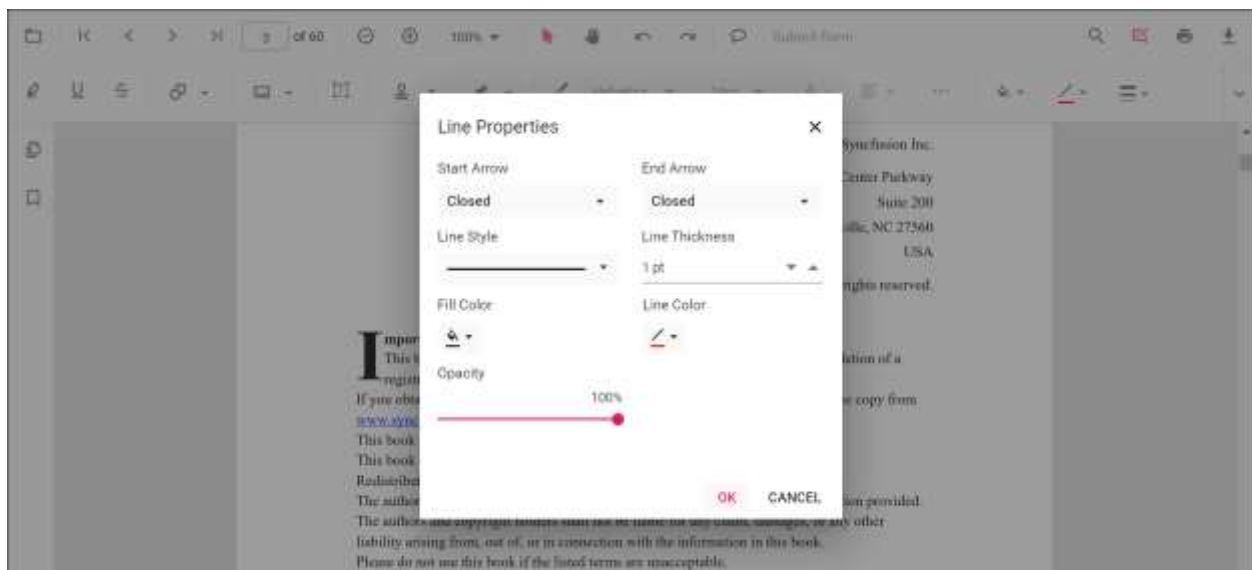
Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Editing the line properties

The properties of the line shapes such as distance and perimeter annotations can be edited using the Line properties window. It can be opened by selecting the Properties option in the context menu that appears on right-clicking the distance and perimeter annotations.



Setting default properties during control initialization

The properties of the shape annotations can be set before creating the control using DistanceSettings, PerimeterSettings, AreaSettings, RadiusSettings, and VolumeSettings.

Refer to the following code sample to set the default annotation settings.

STANDALONE

```

`html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").DistanceSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerDistanceSettings { FillColor = "blue",
Opacity = 0.6, StrokeColor = "green" }).PerimeterSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerPerimeterSettings { FillColor = "green",
Opacity = 0.6, StrokeColor = "blue" }).AreaSettings(new

```



```

Syncfusion.EJ2.PdfViewer.PdfViewerAreaSettings { FillColor = "yellow",
Opacity = 0.6, StrokeColor = "orange" }).RadiusSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerRadiusSettings { FillColor = "orange",
Opacity = 0.6, StrokeColor = "pink" }).VolumeSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerVolumeSettings { FillColor = "pink",
Opacity = 0.6, StrokeColor = "yellow" }).Render()
</div>
` ``

```

SERVER-BACKED

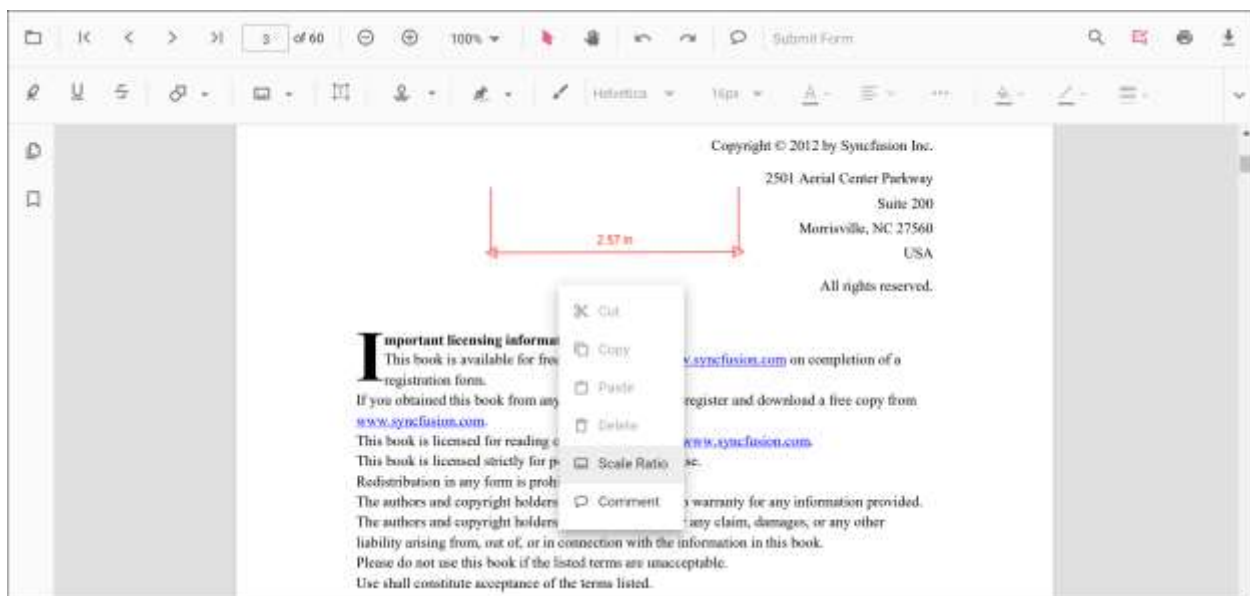
```

` ``html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").DistanceSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerDistanceSettings { FillColor = "blue",
Opacity = 0.6, StrokeColor = "green" }).PerimeterSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerPerimeterSettings { FillColor = "green",
Opacity = 0.6, StrokeColor = "blue" }).AreaSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerAreaSettings { FillColor = "yellow",
Opacity = 0.6, StrokeColor = "orange" }).RadiusSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerRadiusSettings { FillColor = "orange",
Opacity = 0.6, StrokeColor = "pink" }).VolumeSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerVolumeSettings { FillColor = "pink",
Opacity = 0.6, StrokeColor = "yellow" }).Render()
</div>
` ``

```

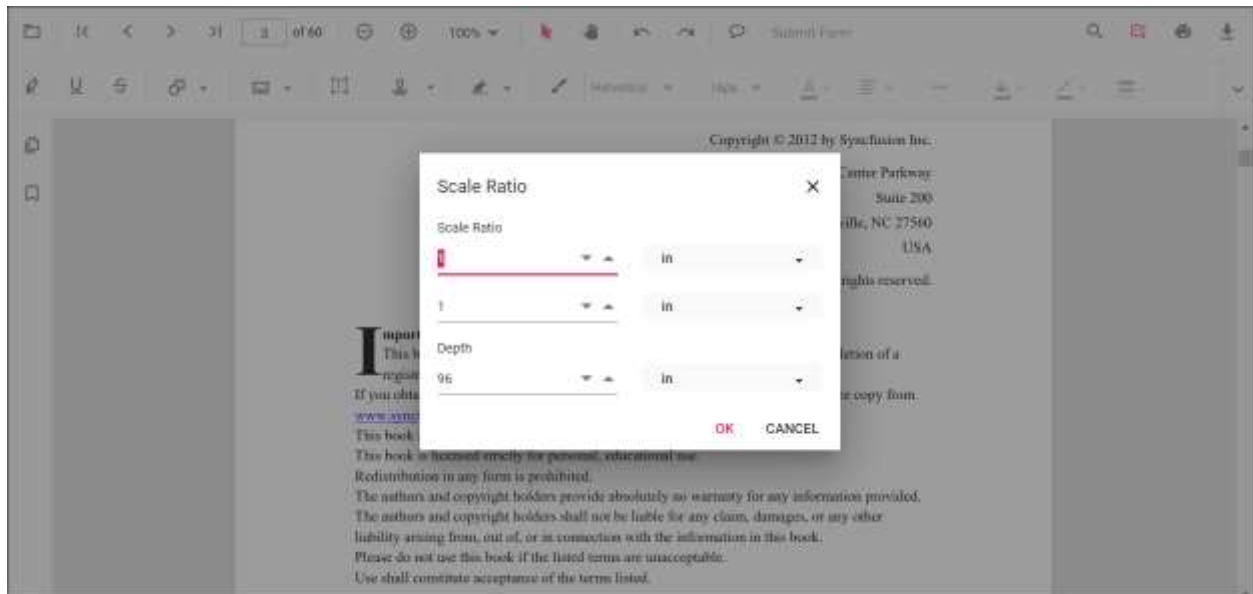
Editing scale ratio and unit of the measurement annotation

The scale ratio and unit of measurement can be modified using the scale ratio option provided in the context menu of the PDF Viewer control.



The Units of measurements supported for the measurement annotations in the PDF Viewer are:

- Inch
- Millimeter
- Centimeter
- Point
- Pica
- Feet



Setting default scale ratio settings during the control initialization

The properties of scale ratio for measurement annotation can be set before creating the control using the ScaleRatioSettings as shown in the following code sample.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").MeasurementSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerMeasurementSettings { ScaleRatio = 2,
ConversionUnit = Syncfusion.EJ2.PdfViewer.CalibrationUnit.Cm }).Render()
</div>
<<<

```

SERVER-BACKED

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").MeasurementSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerMeasurementSettings { ScaleRatio = 2,
ConversionUnit = Syncfusion.EJ2.PdfViewer.CalibrationUnit.Cm }).Render()
</div>
<<<

```

Free Text Annotation in the ASP.NET MVC PDF Viewer component

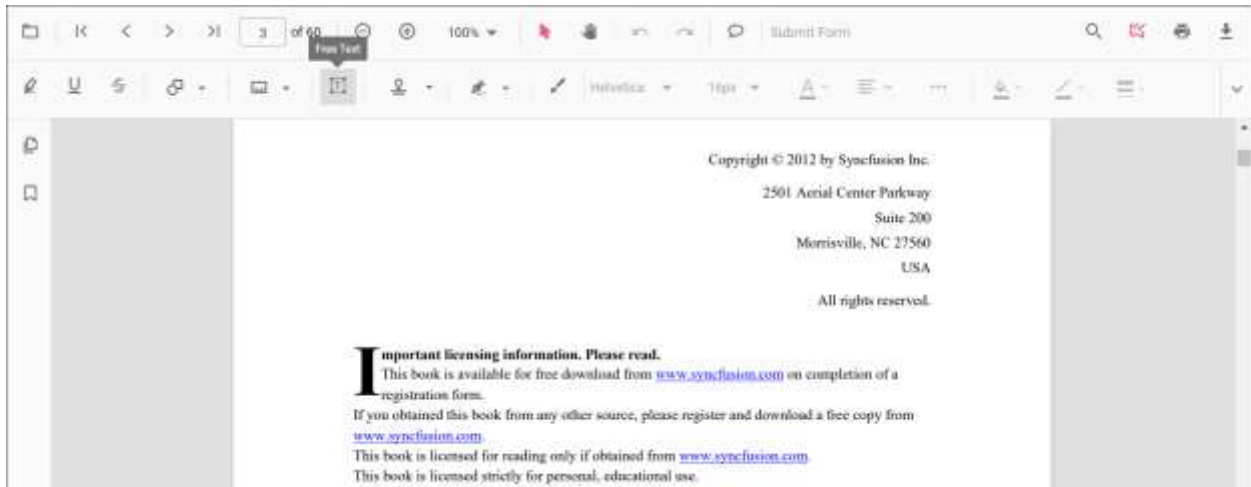
The PDF Viewer control provides the options to add, edit, and delete the free text annotations.

Adding a free text annotation to the PDF document

The Free text annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Free Text Annotation** button in the annotation toolbar. It enables the Free Text annotation mode.
- You can add the text over the pages of the PDF document.

In the pan mode, if the free text annotation mode is entered, the PDF Viewer control will switch to text select mode.



Refer to the following code sample to switch to the Free Text annotation mode.

STANDALONE

```

<html>
<!--Element to set free text annotation mode-->
<button id="set" onclick="addAnnot()">FreeText</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('FreeText');
}
</script>

```

SERVER-BACKED

```

<html>
<!--Element to set free text annotation mode-->
<button id="set" onclick="addAnnot()">FreeText</button>
<div style="width:100%;height:600px">

```

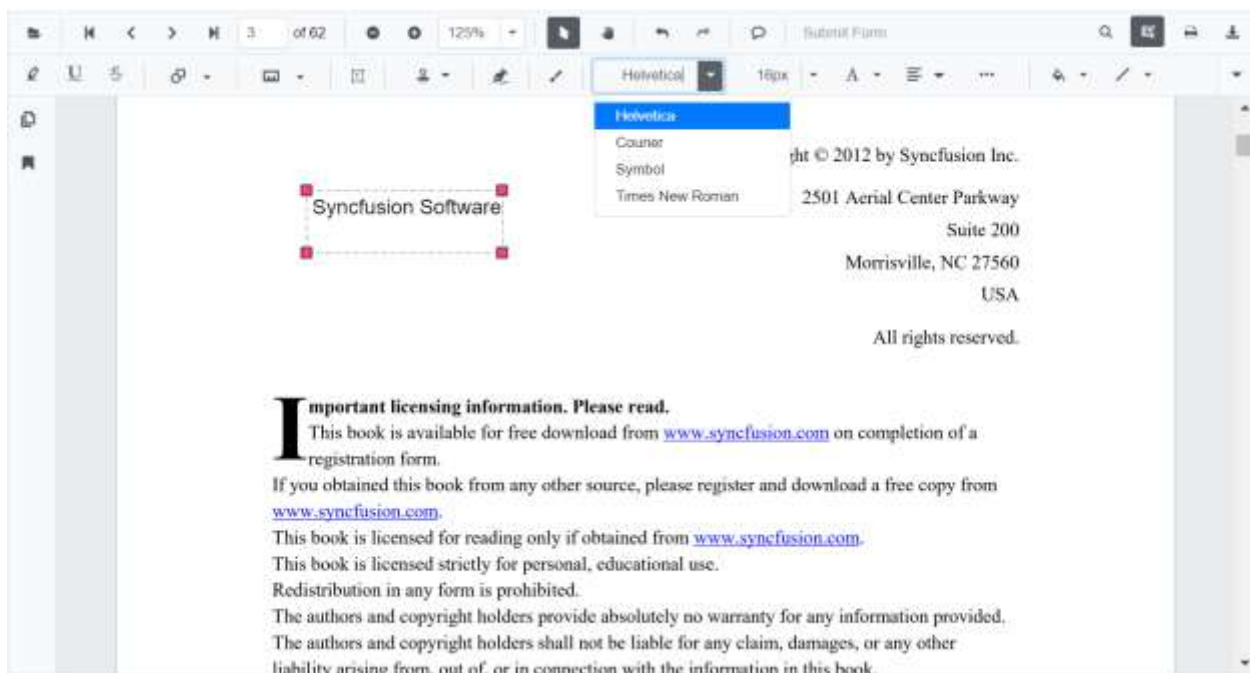
```
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('FreeText');
}
</script>
....
```

Editing the properties of free text annotation

The font family, font size, font styles, font color, text alignment, fill color, the border stroke color, border thickness, and opacity of the free text annotation can be edited using the Font Family tool, Font Size tool, Font Color tool, Text Align tool, Font Style tool, Edit Color tool, Edit Stroke Color tool, Edit Thickness tool, and Edit Opacity tool in the annotation toolbar.

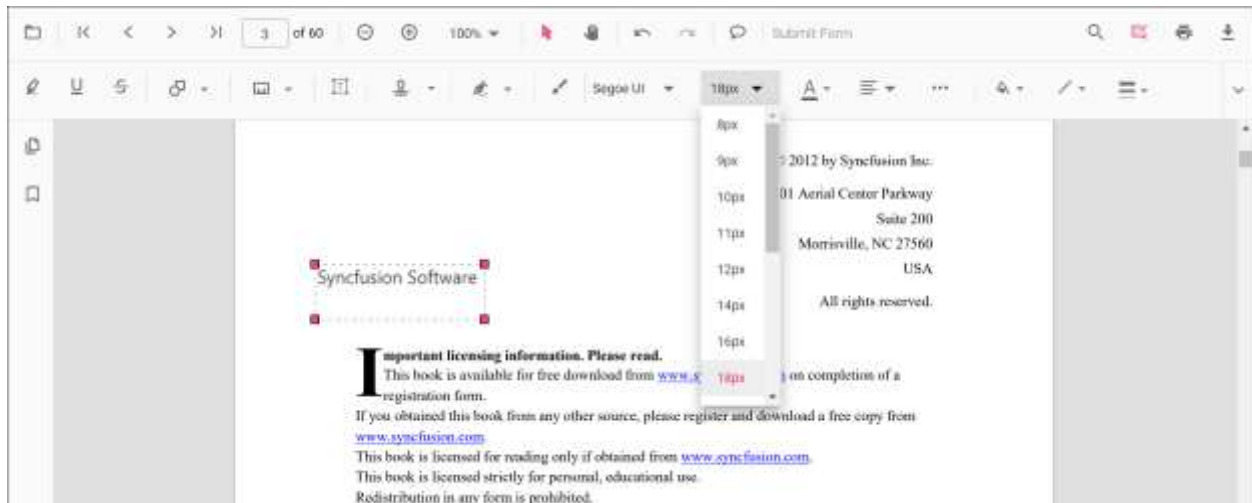
Editing font family

The font family of the annotation can be edited by selecting the desired font in the Font Family tool.



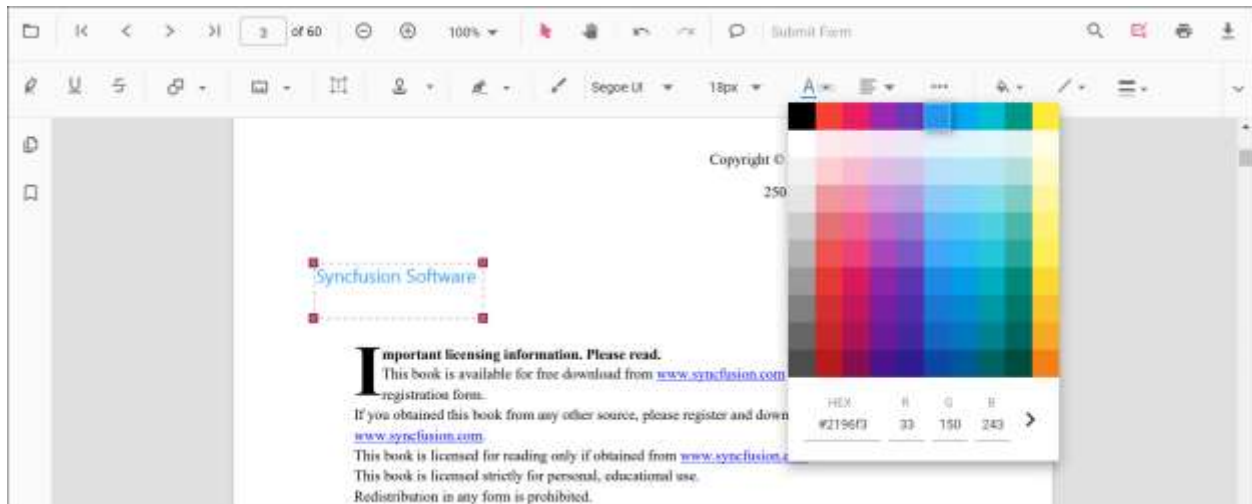
Editing font size

The font size of the annotation can be edited by selecting the desired size in the Font Size tool.



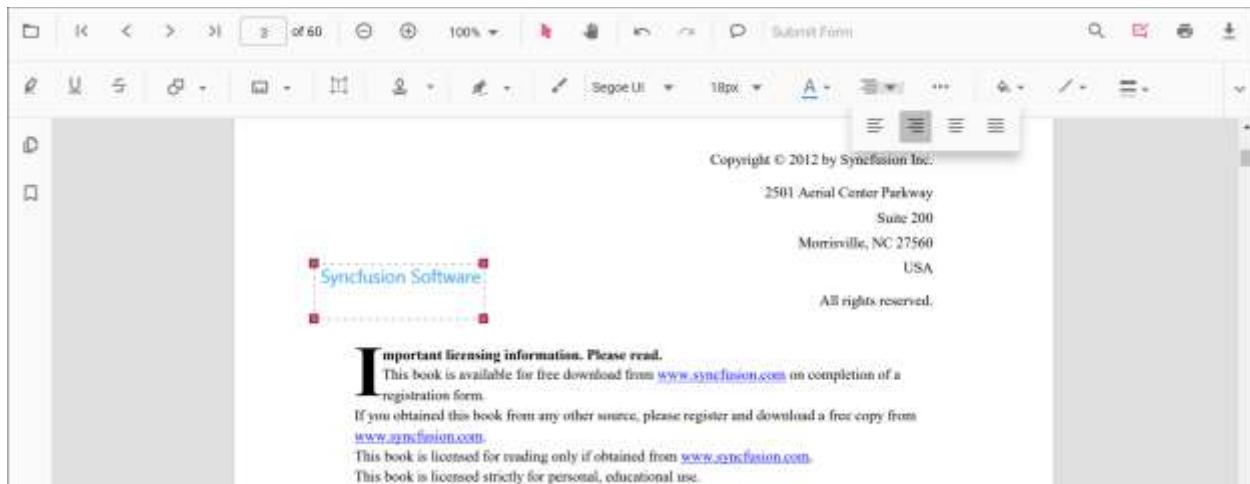
Editing font color

The font color of the annotation can be edited using the color palette provided in the Font Color tool.



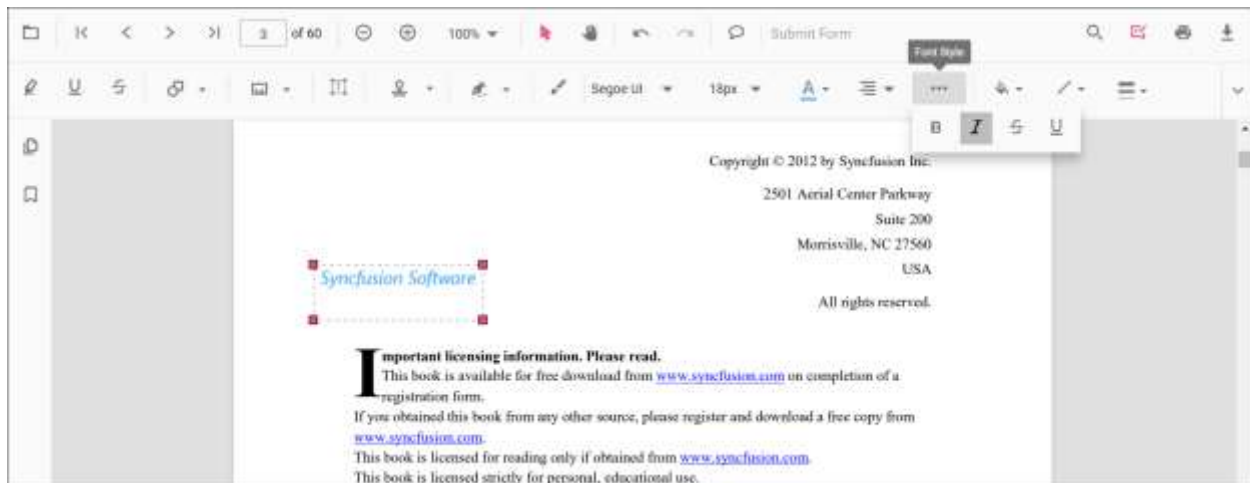
Editing the text alignment

The text in the annotation can be aligned by selecting the desired styles in the drop-down pop-up in the Text Align tool.



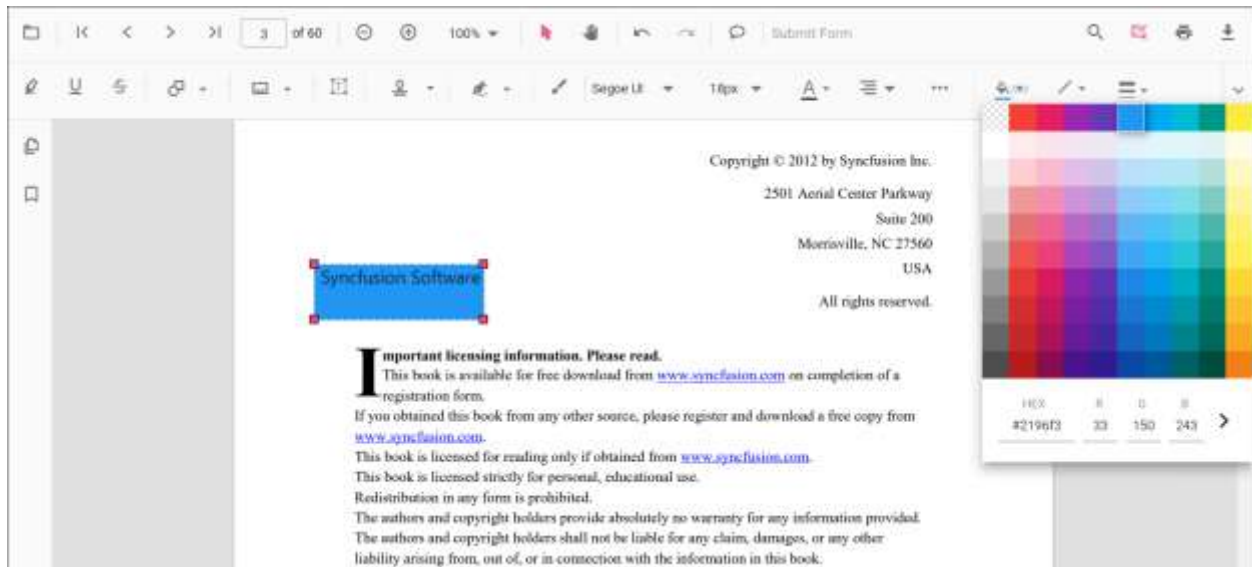
Editing text styles

The style of the text in the annotation can be edited by selecting the desired styles in the drop-down pop-up in the Font Style tool.



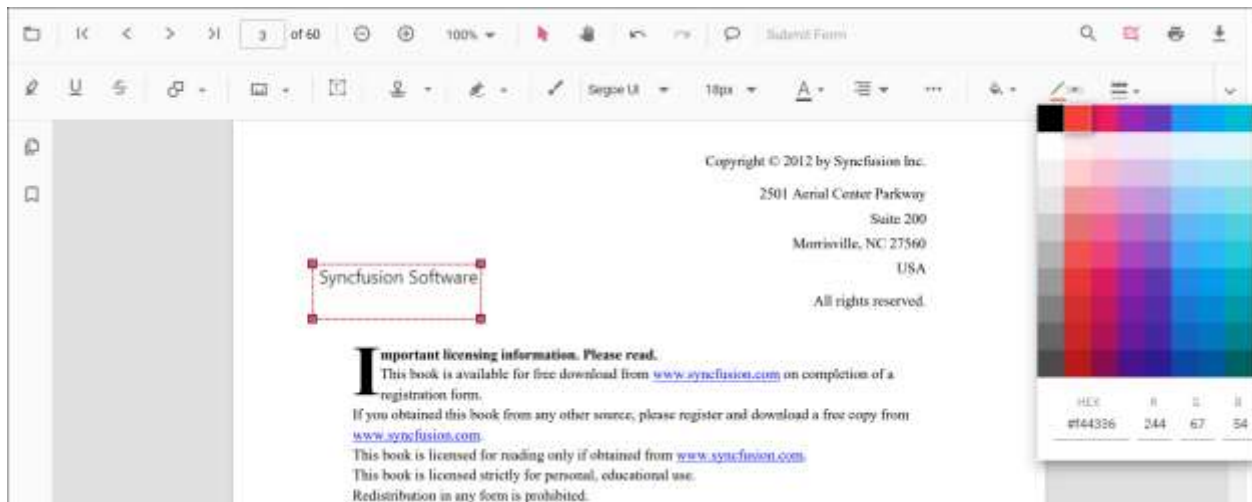
Editing fill color

The fill color of the annotation can be edited using the color palette provided in the Edit Color tool.



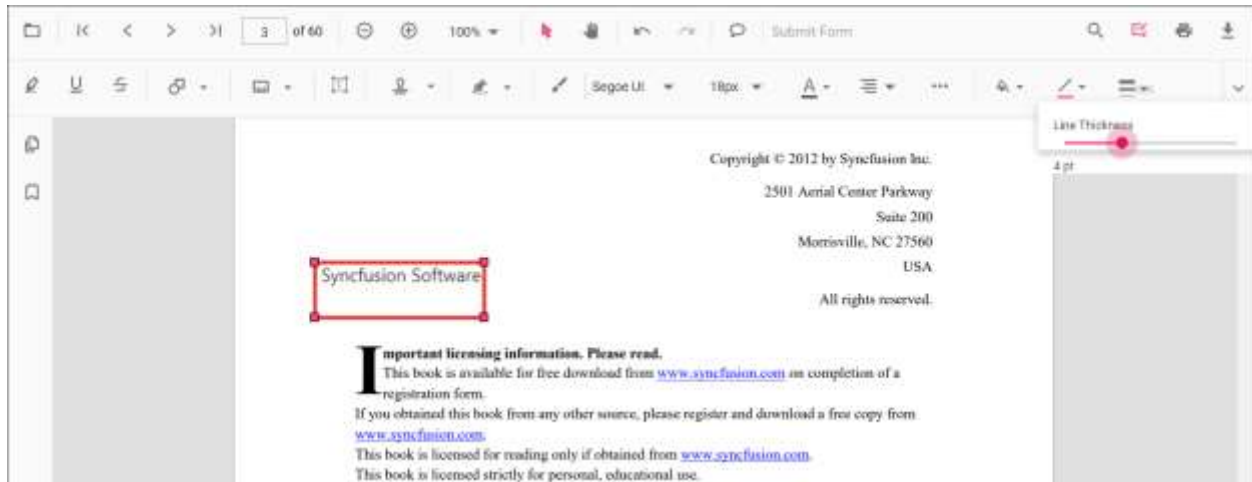
Editing stroke color

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



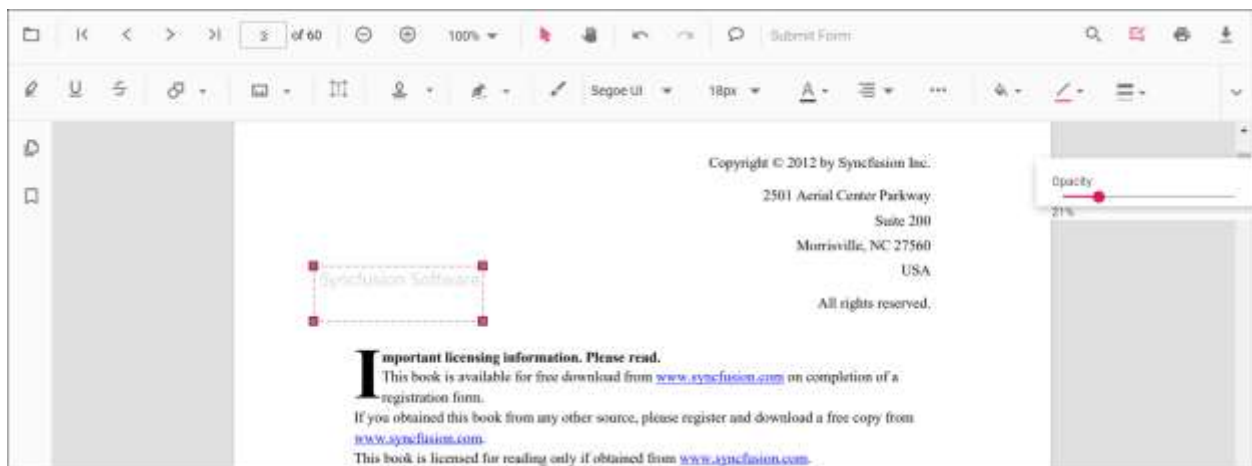
Editing thickness

The border thickness of the annotation can be edited using the range slider provided in the Edit Thickness tool.



Editing opacity

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.



Setting default properties during control initialization

The properties of the free text annotation can be set before creating the control using the FreeTextSettings.

After editing the default values, they will be changed to the selected values.

Refer to the following code sample to set the default free text annotation settings.

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.synCFusion.com/
content/pdf/pdf-succinctly.pdf").FreeTextSettings(new
SynCFusion.EJ2.PdfViewer.PdfViewerFreeTextSettings { FillColor = "green",
BorderColor = "blue", FontColor = "yellow" }).Render()
</div>
<<<

```

SERVER-BACKED

```

<<<html

```



```
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").FreeTextSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerFreeTextSettings { FillColor = "green",
BorderColor = "blue", FontColor = "yellow" }).Render()
</div>
```
```

You can also enable the autofit support for free text annotation by using the `EnableAutoFit` boolean property in `FreeTextSettings` as below. The width of the free text rectangle box will be increased based on the text added to it.

### **STANDALONE**

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").FreeTextSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerFreeTextSettings { EnableAutoFit = true
}).Render()
</div>
```
```

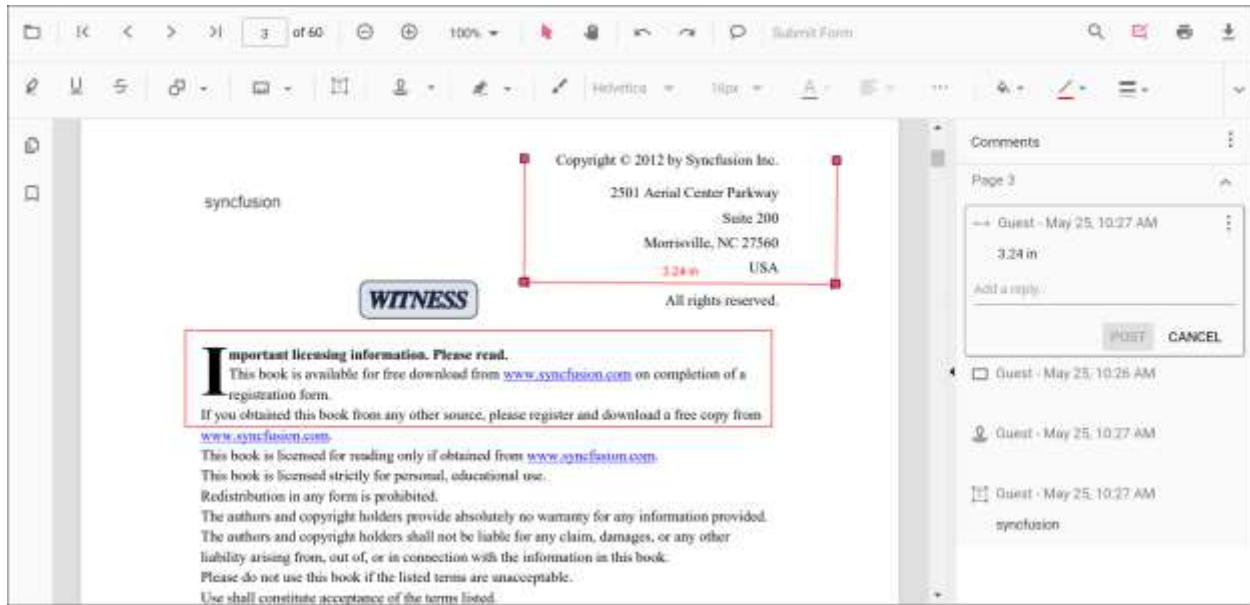
### **SERVER-BACKED**

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").FreeTextSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerFreeTextSettings { EnableAutoFit = true
}).Render()
</div>
```
```

### Comments in the ASP.NET MVC PDF Viewer component

The PDF Viewer control provides options to add, edit, and delete the comments to the following annotation in the PDF documents:

- Shape annotation
- Stamp annotation
- Sticky note annotation
- Measurement annotation
- Text markup annotation
- Free text annotation
- Ink annotation



### Adding a comment to the annotation

Annotation comment, comment replies, and status can be added to the PDF document using the comment panel.

### Comment panel

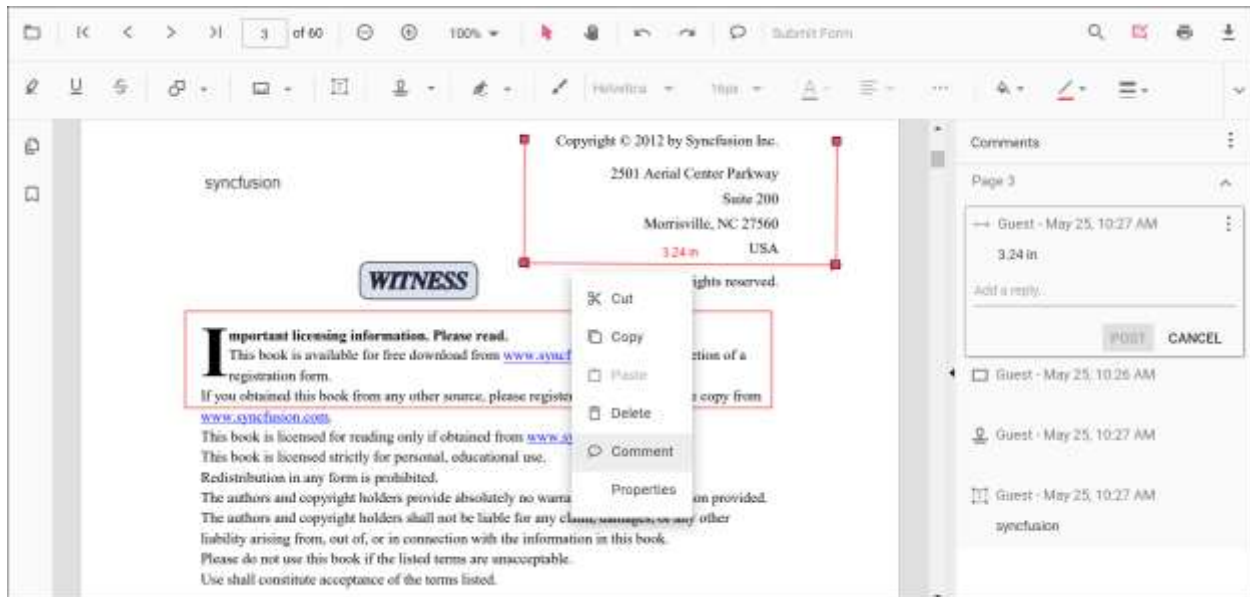
Annotation comments can be added to the PDF using the comment panel. The comment panel can be opened in the following ways:

1. Using the annotation menu
  - Click the Edit Annotation button in the PDF Viewer toolbar. A toolbar appears below it.
  - Click the Comment Panel button. A comment panel will appear.
2. Using Context menu
  - Select annotation in the PDF document and right-click it.
  - Select the comment option in the context menu that appears.
3. Using the Mouse click
  - Select annotation in the PDF document and double click it, a comment panel will appear.

If the comment panel is already in the open state, you can select the annotations and add annotation comments using the comment panel.

### Adding comments

- Select annotation in the PDF document and click it.
- The selected annotation comment container is highlighted in the comment panel.
- Now, you can add comment and comment replies using the comment panel.

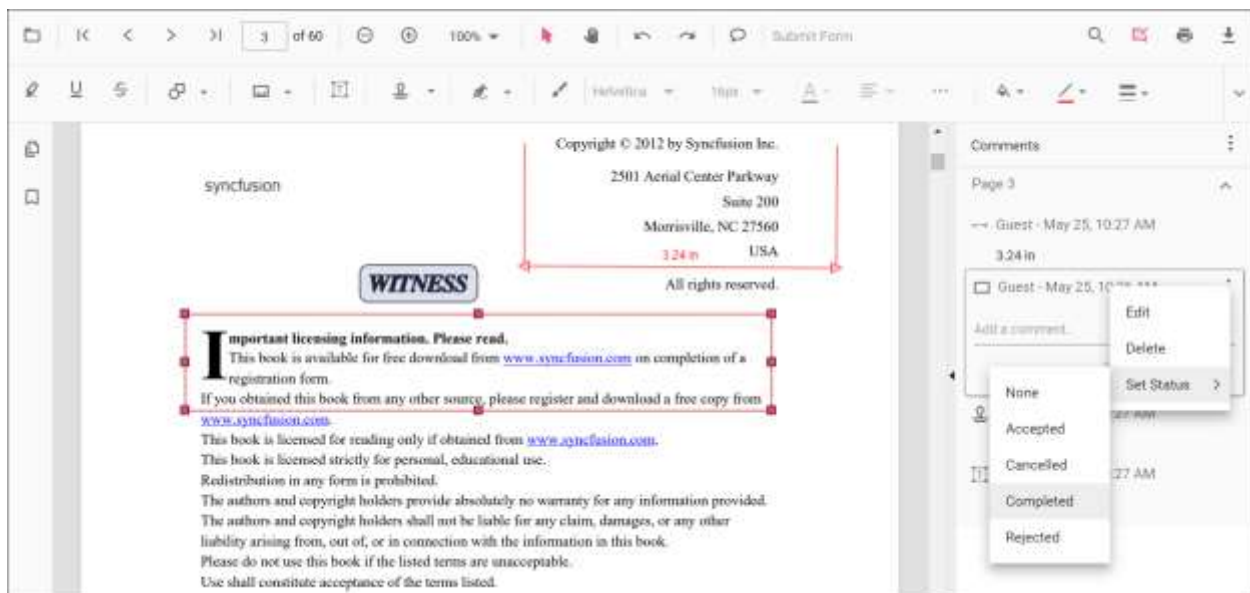


### Adding Comment Replies

- The PDF Viewer control provides an option to add multiple replies to the comment.
- After adding the annotation comment, you can add a reply to the comment.

### Adding Comment or Reply Status

- Select the Annotation Comments in the comment panel.
- Click the more options button showing in the Comments or reply container.
- Select the Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.



### Editing the comments and comments replies of the annotations

The comment, comment replies, and status of the annotation can be edited using the comment panel.

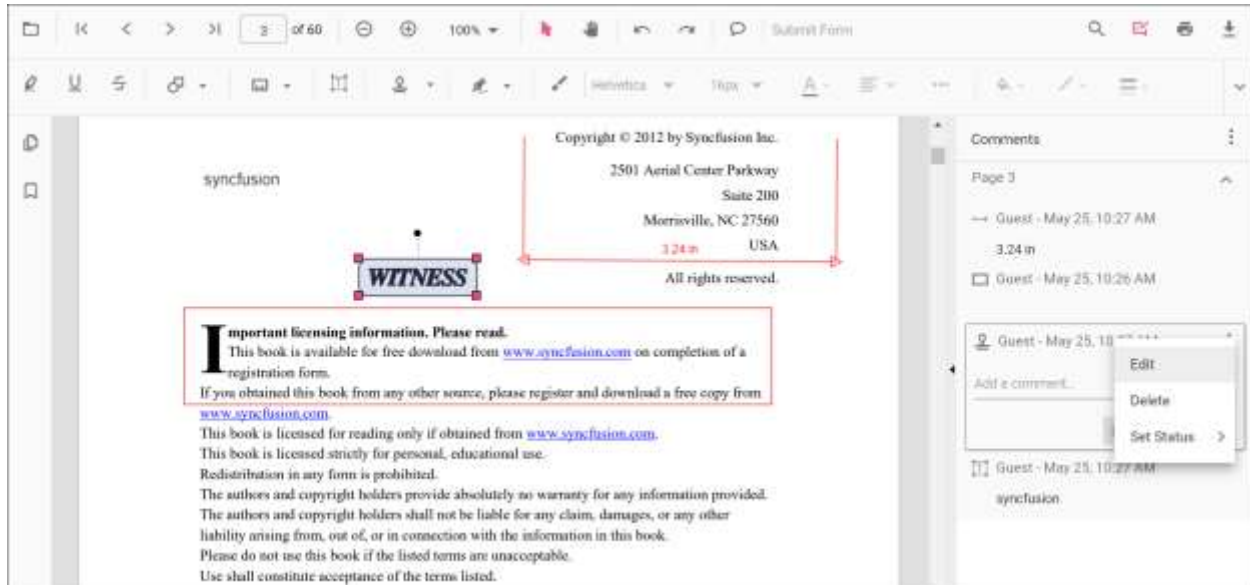
### Editing the Comment or Comment Replies

The annotation comment and comment replies can be edited in the following ways:

1. Using the Context menu
  - Select the Annotation Comments in the comment panel.
  - Click the More options button showing in the Comments or reply container.
  - Select the Edit option in the context menu that appears.
  - Now, an editable text box appears. You can change the content of the annotation comment or comment reply.
2. Using the Mouse Click
  - Select the annotation comments in the comment panel.
  - Double click the comment or comment reply content.
  - Now, an editable text box appears. You can change the content of the annotation comment or comment reply.

### Editing Comment or Reply Status

- Select the Annotation Comments in the comment panel.
- Click the more options button showing in the Comments or reply container.
- Select the Set Status option in the context menu that appears.
- Select the status of the annotation comment in the context menu that appears.
- Status 'None' is the default state. If the status is set to 'None,' the comments or reply does not appear.



### Delete Comment or Comment Replies

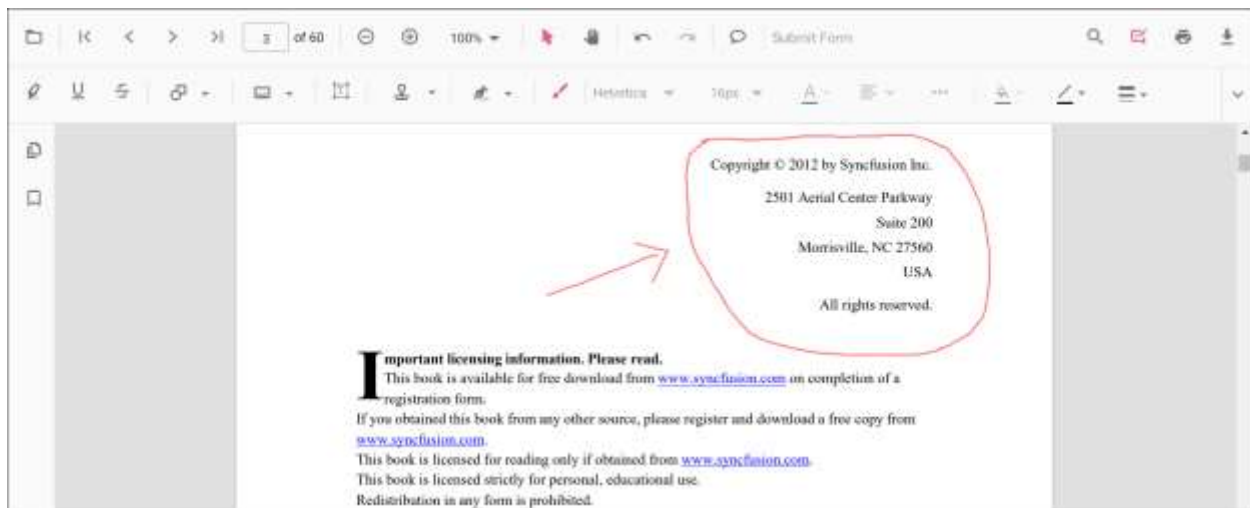
- Select the Annotation Comments in the comment panel.
- Click the more options button shown in the Comments or reply container.
- Select the Delete option in the context menu that appears.



**Note:** The annotation will be deleted on deleting the comment using comment panel.

## Ink Annotation in the ASP.NET MVC PDF Viewer component

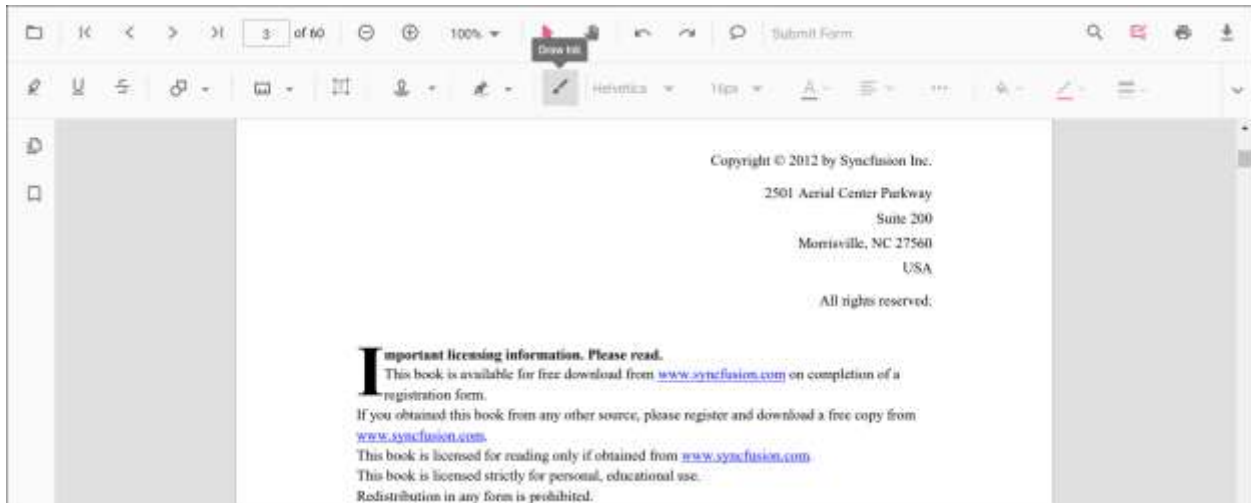
The PDF Viewer control provides the options to add, edit, and delete the ink annotations.



### Adding an ink annotation to the PDF document

The ink annotations can be added to the PDF document using the annotation toolbar.

- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **Draw Ink** button in the annotation toolbar. It enables the ink annotation mode.
- You can draw anything over the pages of the PDF document.



Refer to the following code sample to switch to the ink annotation mode.

### **STANDALONE**

```

```html
<!--Element to set ink annotation mode-->
<button id="set" onclick="addAnnot()">Draw Ink</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Ink');
}
</script>
```

```

### **SERVER-BACKED**

```

```html
<!--Element to set ink annotation mode-->
<button id="set" onclick="addAnnot()">Draw Ink</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
function addAnnot() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.annotation.setAnnotationMode('Ink');
}
</script>
```

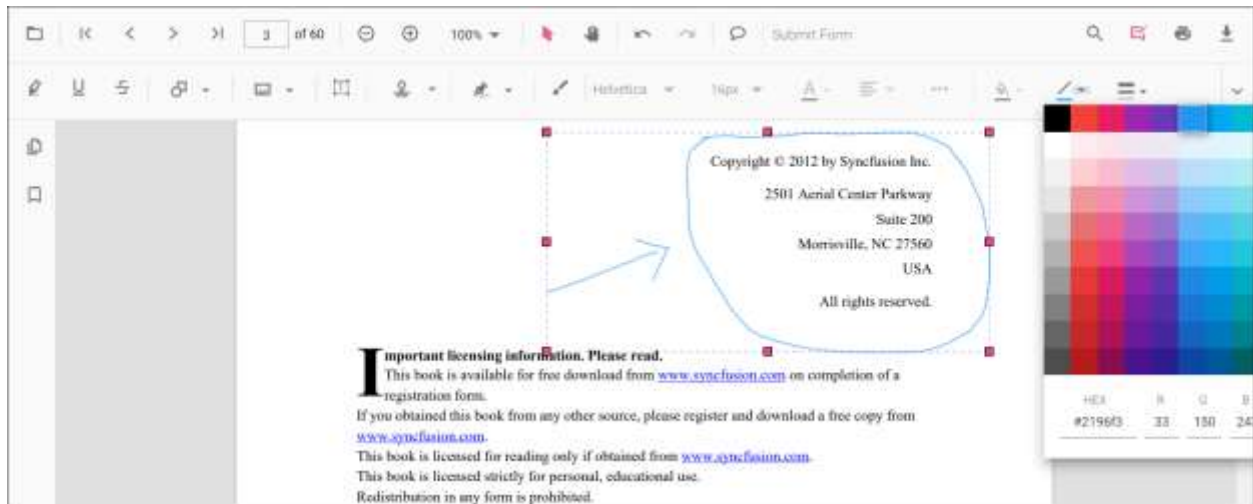
```

### *Editing the properties of the ink annotation*

The stroke color, thickness, and opacity of the ink annotation can be edited using the Edit stroke color tool, Edit thickness tool, and Edit opacity tool in the annotation toolbar.

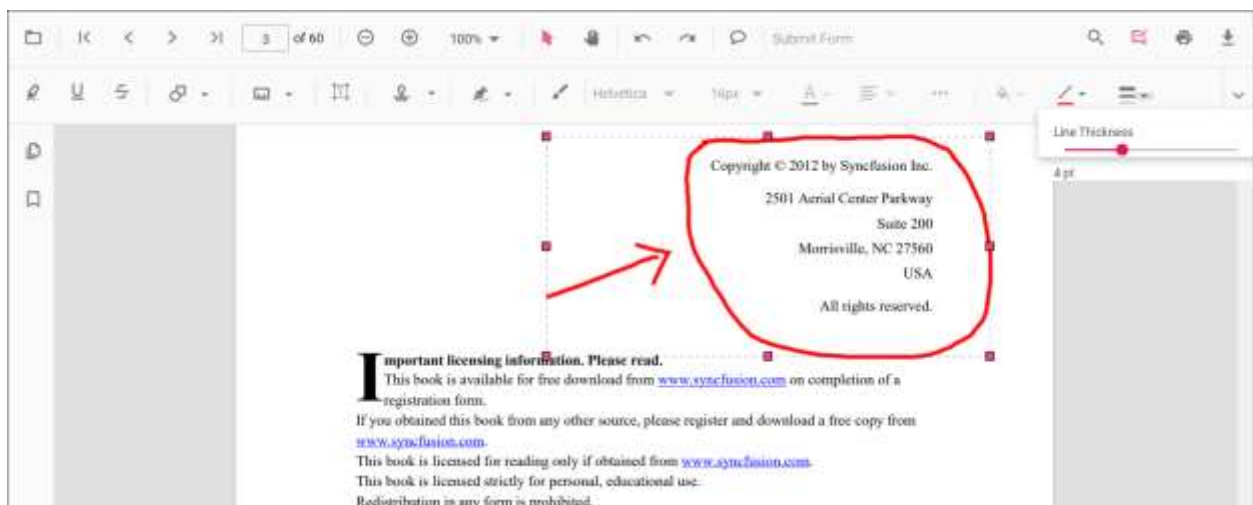
#### *Editing stroke color*

The stroke color of the annotation can be edited using the color palette provided in the Edit Stroke Color tool.



#### *Editing thickness*

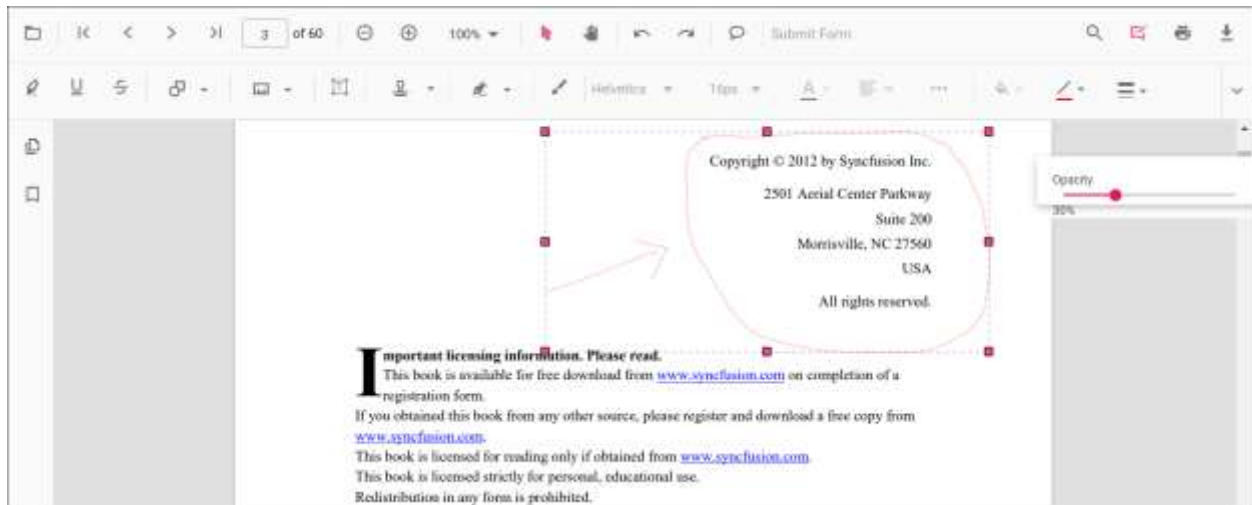
The thickness of the border of the annotation can be edited using the range slider provided in the Edit Thickness tool.



#### *Editing opacity*

The opacity of the annotation can be edited using the range slider provided in the Edit Opacity tool.





## Form Filling

PDF Viewer component allows you to display the form fields available in the PDF document. By using this, you can edit and download the form fields.

The form fields displayed in the PDF Viewer are:

- Text box
- Password box
- Combo box
- Check box
- Radio Button
- Signature Field
- List box

![FormFilling](../../pdfviewer/images/formfilling.png)

## Disabling form fields

The PDF Viewer control provides an option to disable the form fields feature. The code sample for disabling the form fields is as follows.

### STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableFormFields(false).DocumentPath("http://cdn.syncfusion.com/content/pdf/form-filling-document.pdf").Render()
</div>
<<<

```

### SERVER-BACKED

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).EnableFormFields(false).DocumentPath("https://cdn.syncfusion.com/content/pdf/form-filling-document.pdf").Render()
</div>

```





### How to draw handwritten signature in the signature field

Signature can be added to the Signature field by using the following steps:

- Click the Signature Field in the PDF document. The signature panel will appear.

```
![SignatureField](../../pdfviewer/images/signaturefield.png)
```

- Draw the signature in the signature panel.

```
![SignaturePanel](../../pdfviewer/images/signature.png)
```

- Click the **CREATE** button, the drawn signature will be added in the signature field.

```
![Signature](../../pdfviewer/images/sign.png)
```

### Delete the signature inside the signature field

You can also delete the signature in the signature field by using Delete Option in the annotation toolbar.

```
![DeleteSign](../../pdfviewer/images/deletesign.png)
```

### Import and export form fields

The PDF Viewer control provides the support to import and export form fields using a JSON object in the PDF document.

#### Importing form fields using PDF Viewer API

You can import the form fields using JSON file or JSON object in code behind like the below code sample.

```
`html
<button id="viewer" onclick="OnImportFormFieldsClick()">Import FormFields</button>
<div style="width:100%;height:600px">
 @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/form-filling-document.pdf").Render()
</div>
<script>
function OnImportFormFieldsClick() {
 var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
 //The json file has been placed inside the App_Data folder.;
 pdfViewer.importFormFields("D:/PDFViewer/Examples/mvc/sample/App_Data/ImportFormFields.json");
}
</script>
```

**Note:** The JSON file for importing the form fields should be placed in the desired location and the path should be provided correctly.

### Exporting form fields from the PDF document using PDF Viewer API

You can export the form fields as JSON file in code behind as the following code sample.

#### **STANDALONE**

```

<<<html
<button id="viewer" onclick="OnExportFormFieldsClick()">Export
FormFields</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/form-filling-document.pdf").Render()
</div>
<script>
function OnExportFormFieldsClick() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.exportFormFields(null, 'Json');
}
</script>
>>>

```

#### **SERVER-BACKED**

```

<<<html
<button id="viewer" onclick="OnExportFormFieldsClick()">Export
FormFields</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/form-
filling-document.pdf").Render()
</div>
<script>
function OnExportFormFieldsClick() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.exportFormFields(null, 'Json');
}
</script>
>>>

```

### Handwritten Signature

The PDF Viewer control supports adding handwritten signatures to a PDF document. The handwritten signature reduces the paper work of reviewing the content and verifies it digitally.

The following code snippet describes how to disable the handwritten signature in PDF Viewer.

#### **STANDALONE**

```

<<<html
@Html.EJS().PdfViewer("pdfviewer").EnableHandwrittenSignature(false).Documen
tPath("https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
>>>

```

#### **SERVER-BACKED**

```

<<<html
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).EnableHandwrittenSignature(false).DocumentPath("https://cdn
.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
<<<

```

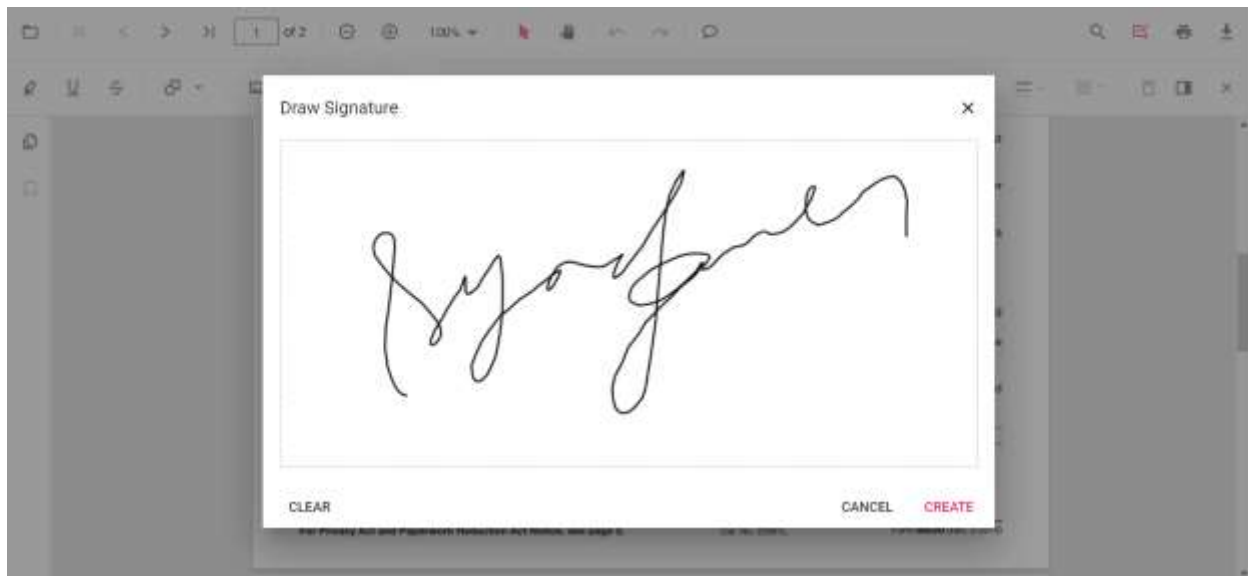
### Adding a handwritten signature to the PDF document

The handwritten signature can be added to the PDF document using the annotation toolbar.

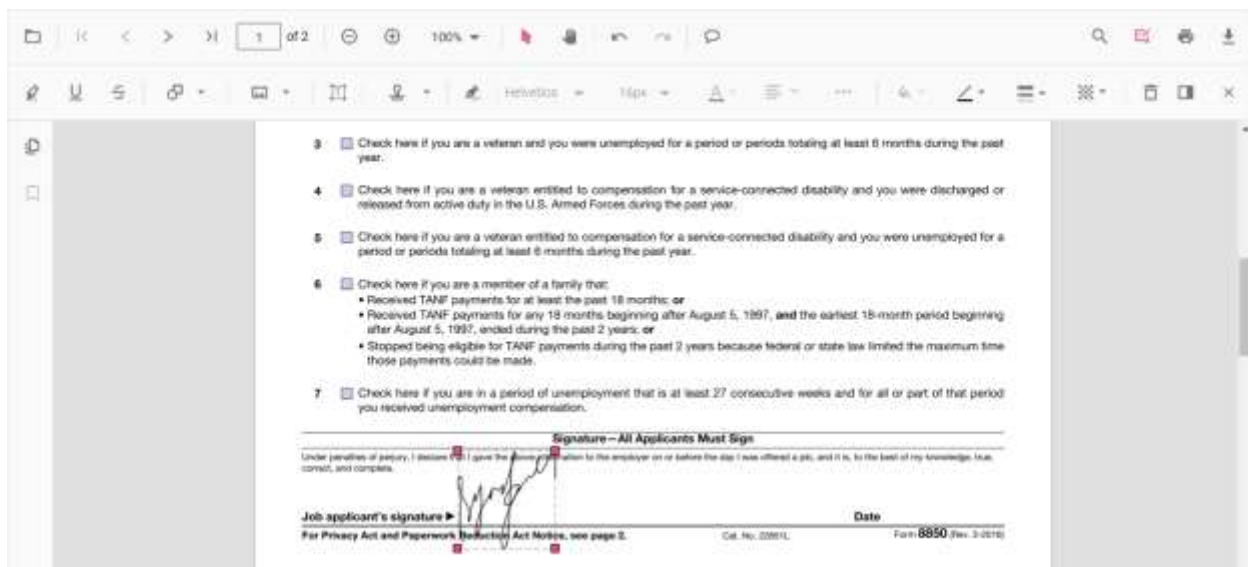
- Click the **Edit Annotation** button in the PDF Viewer toolbar. A toolbar appears below it.
- Select the **HandWritten Signature** button in the annotation toolbar. The signature panel will appear.

The screenshot shows a PDF viewer interface with a toolbar at the top. The main content is a form titled "8850 Pre-Screening Notice and Certification Request for the Work Opportunity Credit". The form is from the Department of the Treasury, Internal Revenue Service, and is dated Rev. March 2016. It includes a "Job applicant: Fill in the lines below and check any boxes that apply. Complete only this side." section with fields for name, social security number, street address, city/town/state/ZIP code, county, and telephone number. Below these fields are two numbered checkboxes for eligibility criteria.

- Draw the signature in the signature panel.

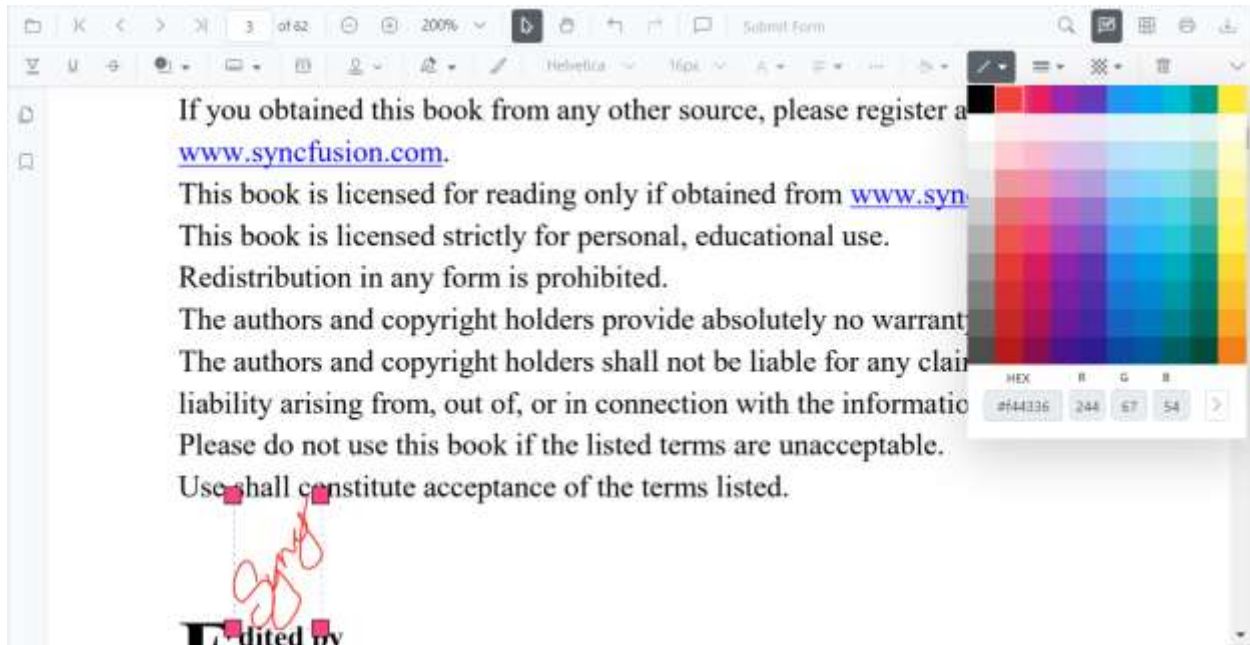


- Then click **Create** button and move the signature using the mouse and place them in the desired location.



### Editing the properties of handwritten signature

The stroke color, border thickness, and opacity of the handwritten signature can be edited using the edit stroke color tool, edit thickness tool, and edit opacity tool in the annotation toolbar.



Refer to the following code snippet to set the default handwritten signature settings.

```
`html
```

```
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).HandWrittenSignatureSettings(new Syncfusion.EJ2.PdfViewer.PdfViewerHandWrittenSignatureSettings { Opacity = 1, Thickness = 2 }).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-succinctly.pdf").Render()
```

See also

- [Toolbar items](#)
- [Feature Modules](#)

## Interaction Mode in PDF Viewer component

The PDF Viewer provides interaction mode for easy interaction with the loaded PDF document. Selection mode and panning mode are the two interaction modes.

### Selection mode

In this mode, the text selection can be performed in the PDF document loaded in PDF Viewer. The panning and scrolling of the pages by touch cannot be performed in this mode. It allows users to select and copy text from the PDF files. This is helpful for copying and sharing text content. You can enable/disable the text selection using the following code snippet.

### STANDALONE

```
```html
<div style="width:100%;height:600px">
  @Html.EJS().PdfViewer("pdfviewer").EnableTextSelection(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

**SERVER-BACKED**

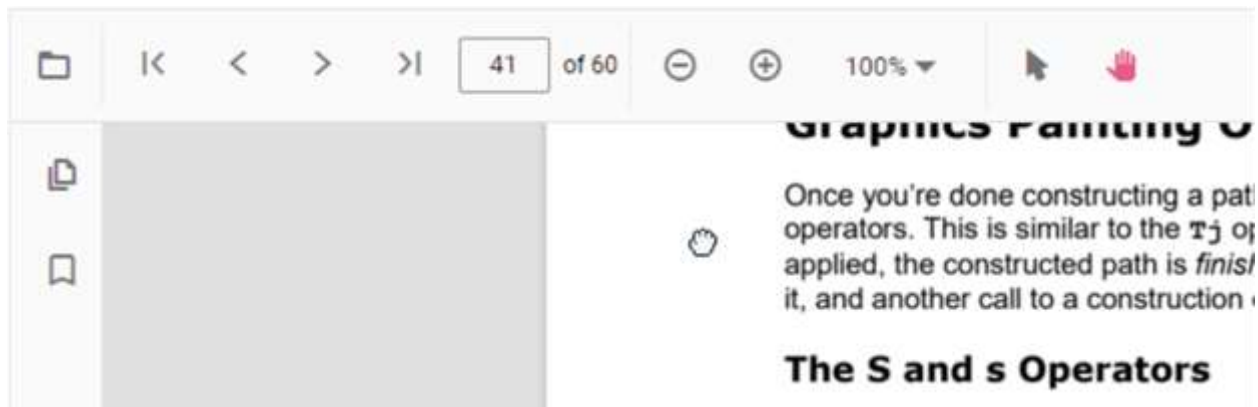
```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).EnableTextSelection(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<<<

```

**Panning Mode**

In this mode, the panning and scrolling of the pages by touch can be performed in the PDF document loaded in the PDF Viewer, but the text selection cannot be performed.



You can switch the interaction mode of PDF Viewer by using the following code snippet.,

**STANDALONE**

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").InteractionMode(Syncfusion.EJ2.PdfViewer.
InteractionMode.Pan).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<<<

```

**SERVER-BACKED**

```

<<<html
<div style="width:100%;height:600px">

```

```
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).InteractionMode(Syncfusion.EJ2.PdfViewer.InteractionMod
e.Pan).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
</div>
```
```

See also

- [Toolbar items](#)
- [Feature Modules](#)

Form Designer

Create form fields programmatically

The PDF Viewer control provides the option to add, edit and delete the Form Fields. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Add a form field to PDF document programmatically

Using addFormField method, the form fields can be added to the PDF document programmatically. We need to pass two parameters in this method. They are Form Field Type and Properties of Form Field Type. To add form field programmatically, Use the following code.

STANDALONE

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/form-
designer.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Rend
er()
</div>
<script>
function documentLoad() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 } });
}
</script>
```
```

SERVER-BACKED

```

```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/fo
rm-
designer.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Rend
er()
</div>
<script>
function documentLoad() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 } });
}
</script>
```

```

Edit/Update form field programmatically

Using updateFormField method, Form Field can be updated programmatically. We should get the Form Field object/Id from FormFieldCollections property that you would like to edit and pass it as a parameter to updateFormField method. The second parameter should be the properties that you would like to update for Form Field programmatically. We have updated the value and background Color properties of Textbox Form Field.

STANDALONE

```

```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/form-
designer.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Rend
er()
</div>
<script>
function documentLoad() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.addFormField("Textbox", { name: "Textfield",
bounds: { X: 300, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.updateFormField(pdfviewer.formFieldCollections[0],
{ backgroundColor: 'red' });
}
</script>
```

```

SERVER-BACKED

```

```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/fo
rm-
designer.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Rend
er()

```



```

</div>
<script>
function documentLoad() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.addFormField("Textbox", { name: "Textfield",
bounds: { X: 300, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.updateFormField(pdfviewer.formFieldCollections[0],
{ backgroundColor: 'red' });
}
</script>
```

```

Delete form field programmatically

Using deleteFormField method, the form field can be deleted programmatically. We should retrieve the Form Field object/Id from FormFieldCollections property that you would like to delete and pass it as a parameter to deleteFormField method. To delete a Form Field programmatically, use the following code.

STANDALONE

```

```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/form-
designer.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Rend
er()
</div>
<script>
function documentLoad() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.addFormField("Textbox", { name: "Textfield",
bounds: { X: 300, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.deleteFormField(pdfviewer.formFieldCollections[0])
;
}
</script>
```

```

SERVER-BACKED

```

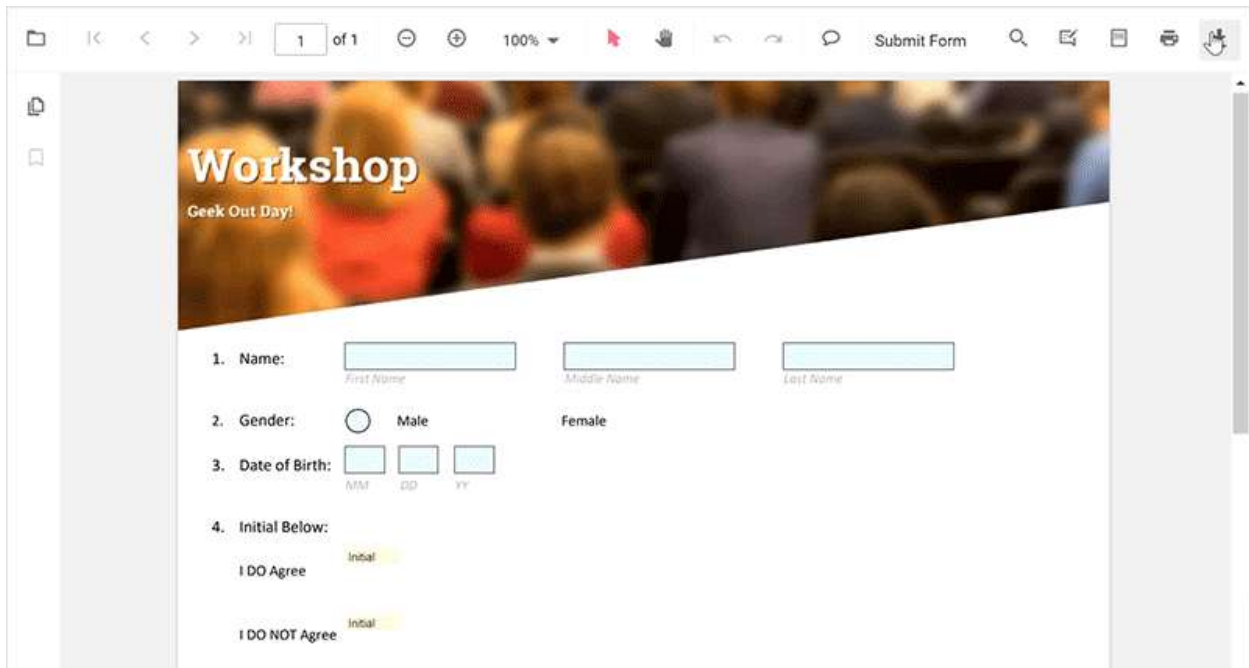
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/fo
rm-
designer.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Rend
er()
</div>
<script>
function documentLoad() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];

```

```
viewer.formDesignerModule.addFormField("Textbox", { name: "Textbox", bounds:
{ X: 146, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.addFormField("Textbox", { name: "Textfield",
bounds: { X: 300, Y: 229, Width: 150, Height: 24 } });
viewer.formDesignerModule.deleteFormField(pdfviewer.formFieldCollections[0])
;
}
</script>
....
```

### *Saving the form fields*

When the download icon is selected on the toolbar, the Form Fields will be saved in the PDF document and this action will not affect the original document. Refer the below GIF for further reference.



You can invoke download action using following code snippet.

### **STANDALONE**

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentLoad("download").EnableDownload(t
true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
</div>
<script>
function download() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.download();
}
</script>
....
```

SERVER-BACKED

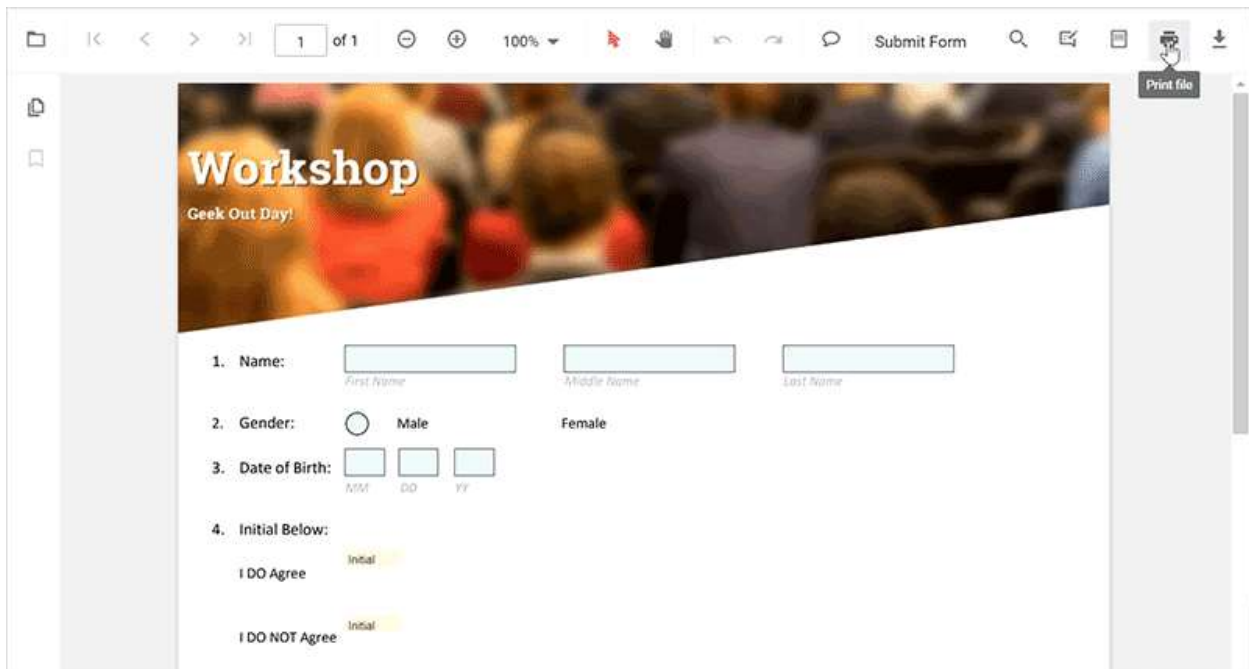
```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentLoad("download").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/")).EnableDownload(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function download() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.download();
}
</script>
<<<

```

Printing the form fields

When the print icon is selected on the toolbar, the PDF document will be printed along with the Form Fields added to the pages and this action will not affect the original document. Refer the below GIF for further reference.



You can invoke print action using the following code snippet.,

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnablePrint(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function print() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.print.print();
}

```

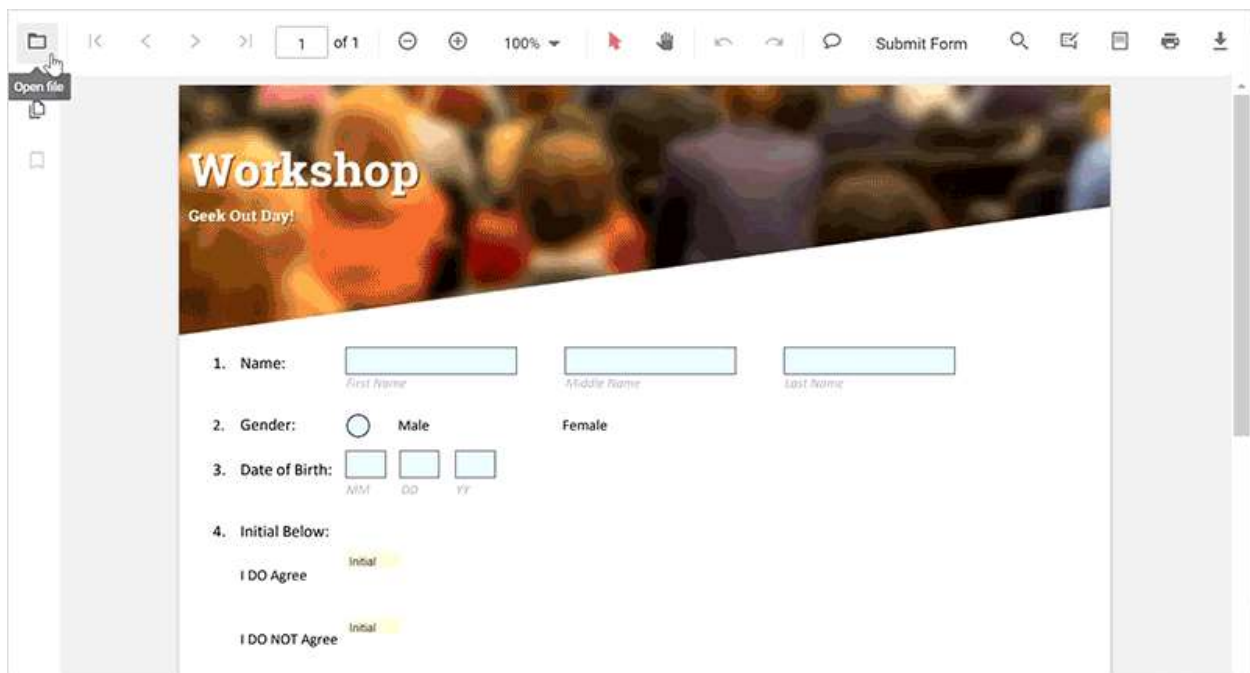
```
</script>
```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).EnablePrint(true).DocumentPath("https://cdn.syncfusion.
com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function print() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.print.print();
}
</script>
```
```

Open the existing PDF document

We can open the already saved PDF document contains Form Fields in it by clicking the open icon in the toolbar. Refer the below GIF for further reference.



Validate form fields

The form fields in the PDF Document will be validated when the `enableFormFieldsValidation` is set to true and hook the `validateFormFields`. The `validateFormFields` will be triggered when the PDF document is downloaded or printed with the non-filled form fields. The non-filled fields will be obtained in the `nonFillableFields` property of the event arguments of `validateFormFields`.

Add the following code snippet to validate the form fields,

STANDALONE

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ValidateFormFields("validateFormFields").
EnableFormFieldsValidation(true).DocumentPath("https://cdn.syncfusion.com/co
ntent/pdf/form-filling-document.pdf").Render()
</div>
<script>
function validateFormFields(args) {
var nonfilledFormFields = args.nonFillableFields;
}
</script>
<<<

```

SERVER-BACKED

```

<<<html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).ValidateFormFields("validateFormFields").EnableFormFieldsVa
lidation(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/form-
filling-document.pdf").Render()
</div>
<script>
function validateFormFields(args) {
var nonfilledFormFields = args.nonFillableFields;
}
</script>
<<<

```

Export and import form fields

The PDF Viewer control provides the support to export and import the form field data in the following formats using the `importFormFields`, `exportFormFields`, and `exportFormFieldsAsObject` methods.

- FDF
- XFDF
- JSON

Export and import as FDF

Using the `exportFormFields` method, the form field data can be exported in the specified data format. This method accepts two parameters:

- The first one must be the destination path for the exported data. If the path is not specified, it will ask for the location while exporting.
- The second parameter should be the format type of the form data.

The following code explains how to export the form field data as FDF.

STANDALONE

```

<<<html
<button id="exportFdf" onclick="exportFdf()">Export FDF</button>
<button id="importFdf" onclick="importFdf()">Import FDF</button>

```

```

<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Export FDF button click.
function exportFdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Fdf);
}
// Event triggers on Import FDF button click.
function importFdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired
location, and the path should be provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Fdf);
}
</script>

```

SERVER-BACKED

```

```html
<button id="exportFdf" onclick="exportFdf()">Export FDF</button>
<button id="importFdf" onclick="importFdf()">Import FDF</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Export FDF button click.
function exportFdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Fdf);
}
// Event triggers on Import FDF button click.
function importFdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired
location, and the path should be provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Fdf);
}
</script>
```

```

Export and import as XFDF

The following code explains how to export the form field data as XFDF.

STANDALONE

```

```html
<button id="exportXfdf" onclick="exportXfdf()">Export XFDF</button>
<button id="importXfdf" onclick="importXfdf()">Import XFDF</button>

```

```

<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Export XFDF button click.
function exportXfdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Xfdf);
}
// Event triggers on Import XFDF button click.
function importXfdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired
location, and the path should be provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Xfdf);
}
</script>

```

### **SERVER-BACKED**

```

```html
<button id="exportXfdf" onclick="exportXfdf()">Export XFDF</button>
<button id="importXfdf" onclick="importXfdf()">Import XFDF</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Export XFDF button click.
function exportXfdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Xfdf);
}
// Event triggers on Import XFDF button click.
function importXfdf() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired
location, and the path should be provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Xfdf);
}
</script>
```

```

### Export and import as JSON

The following code explains how to export the form field data as JSON.

### **STANDALONE**

```

```html
<button id="exportJson" onclick="exportJson()">Export JSON</button>
<button id="importJson" onclick="importJson()">Import JSON</button>

```

```

<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Export JSON button click.
function exportJson() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Json);
}
// Event triggers on Import JSON button click.
function importJson() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired
location, and the path should be provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Json);
}
</script>

```

SERVER-BACKED

```

<<<html
<button id="exportJson" onclick="exportJson()">Export JSON</button>
<button id="importJson" onclick="importJson()">Import JSON</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Export JSON button click.
function exportJson() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Data must be the desired path for the exported document.
viewer.exportFormFields('Data', FormFieldDataFormat.Json);
}
// Event triggers on Import JSON button click.
function importJson() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// The file for importing the form fields should be placed in the desired
location, and the path should be provided correctly.
viewer.importFormFields('File', FormFieldDataFormat.Json);
}
</script>

```

Export and import as Object

The PDF Viewer control supports exporting the form field data as an object, and the exported data will be imported into the current PDF document from the object.

The following code shows how to export the form field data as an object and import the form field data from that object into the current PDF document via a button click.

STANDALONE

```

<html>
<button id="exportDataAsObject" onclick="exportDataAsObject()">Export
Object</button>
<button id="importData" onclick="importData()">Import Data</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
var exportedData;
// Event triggers on Export Object button click.
function exportDataAsObject() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Export the form field data to an FDF object.
viewer.exportFormFieldsAsObject(FormFieldDataFormat.Fdf).then(value => {
exportedData = value;
})
//// Export the form field data to an XFDF object.
//viewer.exportFormFieldsAsObject(FormFieldDataFormat.Xfdf).then(value => {
//    exportedData = value;
//})
//// Export the form field data to an JSON object.
//viewer.exportFormFieldsAsObject(FormFieldDataFormat.Json).then(value => {
//    exportedData = value;
//})
}
// Event triggers on Import Data button click.
function importData() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Import the form field data from the FDF object into the current PDF
document.
viewer.importFormFields(exportedData, FormFieldDataFormat.Fdf);
//// Import the form field data from the XFDF object into the current PDF
document.
//viewer.importFormFields (exportedData, FormFieldDataFormat.Xfdf);
//// Import the form field data from the FDF object into the current PDF
document.
//viewer.importFormFields (exportedData, FormFieldDataFormat.Json);
}
</script>
</html>

```

SERVER-BACKED

```

<html>
<button id="exportDataAsObject" onclick="exportDataAsObject()">Export
Object</button>
<button id="importData" onclick="importData()">Import Data</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>

```

```

var exportedData;
// Event triggers on Export Object button click.
function exportDataAsObject() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Export the form field data to an FDF object.
viewer.exportFormFieldsAsObject(FormFieldDataFormat.Fdf).then(value => {
exportedData = value;
})
//// Export the form field data to an XFDF object.
//viewer.exportFormFieldsAsObject(FormFieldDataFormat.Xfdf).then(value => {
//    exportedData = value;
//})
//// Export the form field data to an JSON object.
//viewer.exportFormFieldsAsObject(FormFieldDataFormat.Json).then(value => {
//    exportedData = value;
//})
}
// Event triggers on Import Data button click.
function importData() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// Import the form field data from the FDF object into the current PDF
document.
viewer.importFormFields(exportedData, FormFieldDataFormat.Fdf);
//// Import the form field data from the XFDF object into the current PDF
document.
//viewer.importFormFields (exportedData, FormFieldDataFormat.Xfdf);
//// Import the form field data from the FDF object into the current PDF
document.
//viewer.importFormFields (exportedData, FormFieldDataFormat.Json);
}
</script>

```

Form Field Properties

Form field properties in Syncfusion PDF Viewer allow you to customize and interact with form fields embedded within PDF documents. This documentation provides an overview of the form field properties supported by the Syncfusion PDF Viewer and explains how to use them effectively.

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Signature and initial fields settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the signature field properties on a button click.

STANDALONE

```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Initial',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Initial',
thickness: 4
});
}
</script>

```

SERVER-BACKED

```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Initial',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Initial',
thickness: 4
});
}
</script>

```

The following code example explains how to update the properties of the signature field added to the document from the form designer toolbar.

```

<script type="text/javascript">
window.onload = function () {

```

```

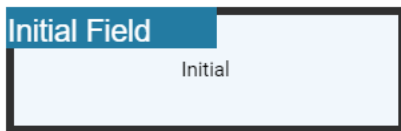
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.signatureFieldSettings = {
// Set the name of the form field element.
name: 'Signature',
// Specify whether the signature field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the signature field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'Signature',
// Set the thickness of the signature field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Specify the properties of the signature indicator in the signature field.
signatureIndicatorSettings: {
opacity: 1,
backgroundColor: '#daeaf7ff',
height: 50,
fontSize: 15,
text: 'Signature Field',
color: 'white'
}
}
}
</script>

```



The following code example explains how to update the properties of the initial field added to the document from the form designer toolbar.

```
<script type="text/javascript">
window.onload = function () {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.initialFieldSettings = {
// Set the name of the form field element.
name: 'Initial',
// Specify whether the initial field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the initial field.
isPrint: true,
// Set the text to be displayed as tooltip.
tooltip: 'Initial',
// Set the thickness of the initial field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Specify the properties of the initial indicator in the initial field.
initialIndicatorSettings: {
opacity: 1,
backgroundColor: '#daeaf7ff',
height: 50,
fontSize: 15,
text: 'Initial Field',
color: 'white'
},
}
}
</script>
```



Textbox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the Textbox field properties on a button click.

STANDALONE

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Textbox',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Textbox',
thickness: 4,
value: 'Textbox',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
isMultiline: false,
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }
});
}
</script>
```

SERVER-BACKED

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
```

```

<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Textbox',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Textbox',
thickness: 4,
value: 'Textbox',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
isMultiline: false,
bounds: { X: 146, Y: 229, Width: 150, Height: 24 }
});
}
</script>

```

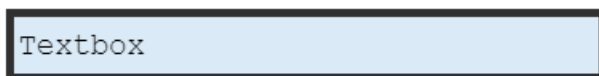
The following code example explains how to update the properties of the textbox field added to the document from the form designer toolbar.

```

<script type="text/javascript">
window.onload = function () {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.textFieldSettings = {
// Set the name of the form field element.
name: 'Textbox',
// Specify whether the Textbox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the Textbox field.
isPrint: true,
// Set the text to be displayed as a tooltip.

```

```
tooltip: 'Textbox',
// Set the thickness of the Textbox field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value: 'Textbox',
// Set the font family of the textbox field.
fontFamily: 'Courier',
// Set the font size of the textbox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the textbox field.
color: 'black',
// Set the border color of the textbox field.
borderColor: 'black',
// Set the background color of the textbox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
// Allows multiline input in the text field. FALSE, by default.
isMultiline: false
}
}
</script>
```



[Password field settings](#)

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the Password field properties on a button click.

STANDALONE


```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("PDF_Succinctly.pdf").Render
()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Password',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Password',
thickness: 4,
value: 'Password',
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
maxLength: 0,
bounds: { X: 148, Y: 229, Width: 150, Height: 24 }
});
}
</script>

```

SERVER-BACKED

```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("PDF_Succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'Password',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'Password',
thickness: 4,
value: 'Password',
fontFamily: 'Courier',

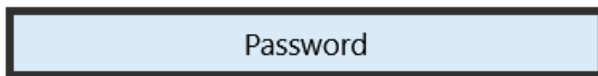
```

```
fontSize: 10,  
fontStyle: 'None',  
color: 'black',  
borderColor: 'black',  
backgroundColor: '#daeaf7ff',  
alignment: 'Left',  
maxLength: 0,  
bounds: { X: 148, Y: 229, Width: 150, Height: 24 }  
});  
}  
</script>
```

The following code example explains how to update the properties of the password field added to the document from the form designer toolbar.

```
<script type="text/javascript">  
window.onload = function () {  
var viewer = document.getElementById('pdfviewer').ej2_instances[0];  
viewer.passwordFieldSettings = {  
// Set the name of the form field element.  
name: 'Password',  
// Specify whether the Password field is in read-only or read-write mode.  
isReadOnly: false,  
// Set the visibility of the form field.  
visibility: 'visible',  
// Specify whether the field is mandatory or not.  
isRequired: false,  
// Specify whether to print the Password field.  
isPrint: true,  
// Set the text to be displayed as a tooltip.  
tooltip: 'Password',  
// Set the thickness of the Password field. To hide the borders, set the value to 0 (zero).  
thickness: 4,  
// Set the value of the form field element.  
value: 'Password',  
// Set the font family of the Password field.  
fontFamily: 'Courier',  
// Set the font size of the Password field.  
fontSize: 10,
```

```
// Specify the font style
fontStyle: 'None',
// Set the font color of the Password field.
color: 'black',
// Set the border color of the Password field.
borderColor: 'black',
// Set the background color of the Password field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the maximum character length.
maxLength: 0,
}
}
</script>
```



CheckBox field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the CheckBox field properties on a button click.

STANDALONE

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'CheckBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'CheckBox',
thickness: 4,
```

```

isChecked: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'CheckBox'
});
}
</script>

```

SERVER-BACKED

```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'CheckBox',
isReadOnly: true,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'CheckBox',
thickness: 4,
isChecked: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'CheckBox'
});
}
</script>

```

The following code example explains how to update the properties of the checkbox field added to the document from the form designer toolbar.

```

<script type="text/javascript">
window.onload = function () {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.checkBoxFieldSettings = {
// Set the name of the form field element.
name: 'CheckBox',
// Specify whether the CheckBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.

```

```

visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the CheckBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'CheckBox',
// Set the thickness of the CheckBox field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Specifies whether the check box is in checked state or not.
isChecked: true,
// Set the background color of the check box in hexadecimal string format.
backgroundColor: '#daeaf7f',
// Set the border color of the check box field.
borderColor: 'black'
// Set the value of the form field element.
value: 'CheckBox'
}
}
</script>

```



RadioButton field settings

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the RadioButton field properties on a button click.

STANDALONE

```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>

```

```

<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'RadioButton',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'RadioButton',
thickness: 4,
isSelected: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'RadioButton'
});
}
}
</script>

```

SERVER-BACKED

```

<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'RadioButton',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'RadioButton',
thickness: 4,
isSelected: true,
backgroundColor: '#daeaf7ff',
borderColor: 'black',
value: 'RadioButton'
});
}
}
</script>

```

The following code example explains how to update the properties of the radiobutton field added to the document from the form designer toolbar.

```
<script type="text/javascript">
```

```
window.onload = function () {  
var viewer = document.getElementById('pdfviewer').ej2_instances[0];  
viewer.radioButtonFieldSettings = {  
  // Set the name of the form field element.  
  name: 'RadioButton',  
  // Specify whether the RadioButton field is in read-only or read-write mode.  
  isReadOnly: false,  
  // Set the visibility of the form field.  
  visibility: 'visible',  
  // Specify whether the field is mandatory or not.  
  isRequired: false,  
  // Specify whether to print the RadioButton field.  
  isPrint: true,  
  // Set the text to be displayed as a tooltip.  
  tooltip: 'RadioButton',  
  // Set the thickness of the RadioButton field. To hide the borders, set the value to 0 (zero).  
  thickness: 4,  
  // Specifies whether the radio button is in checked state or not.  
  isSelected: true,  
  // Set the background color of the radio button in hexadecimal string format.  
  backgroundColor: '#daeaf7ff',  
  // Set the border color of the radio button field.  
  borderColor: 'black'  
  // Set the value of the form field element.  
  value: 'RadioButton'  
}  
}  
</script>
```



[ListBox field settings](#)

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the ListBox field properties on a button click.

STANDALONE

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'ListBox',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'ListBox',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions
});
}
</script>
```

SERVER-BACKED

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
```



```

{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'ListBox',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'ListBox',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions
});
}
</script>

```

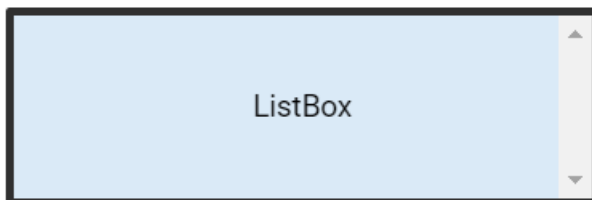
The following code example explains how to update the properties of the listbox field added to the document from the form designer toolbar.

```

<script type="text/javascript">
window.onload = function () {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.listBoxFieldSettings = {
// Set the name of the form field element.
name: 'ListBox',
// Specify whether the ListBox field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',
// Specify whether the field is mandatory or not.
isRequired: false,

```

```
// Specify whether to print the ListBox field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'ListBox',
// Set the thickness of the ListBox field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'ListBox',
// Set the font family of the ListBox field.
fontFamily: 'Courier',
// Set the font size of the ListBox field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the ListBox field.
color: 'black',
// Set the border color of the ListBox field.
borderColor: 'black',
// Set the background color of the ListBox field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the listbox items.
options: customOptions
};
}
</script>
```



[DropDown field settings](#)

Using the `updateFormField` method, the form fields can be updated programmatically.

The following code example explains how to update the DropDown field properties on a button click.

STANDALONE

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formField = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'DropDown',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'DropDown',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions,
});
}
</script>
```

SERVER-BACKED

```
<button id="updateProperties" onclick="updateProperties()">Update
Properties</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
<script>
// Event triggers on Update Properties button click.
function updateProperties() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
```

```

var formField = viewer.retrieveFormFields();
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
var formField = viewer.retrieveFormFields();
viewer.formDesignerModule.updateFormField(formField[0], {
name: 'DropDown',
isReadOnly: false,
visibility: 'visible',
isRequired: false,
isPrint: true,
tooltip: 'DropDown',
thickness: 4,
fontFamily: 'Courier',
fontSize: 10,
fontStyle: 'None',
color: 'black',
borderColor: 'black',
backgroundColor: '#daeaf7ff',
alignment: 'Left',
options: customOptions,
});
}
</script>

```

The following code example explains how to update the properties of the dropdown field added to the document from the form designer toolbar.

```

<script type="text/javascript">
window.onload = function () {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
const customOptions = [
{ itemName: 'item1', itemValue: 'item1' },
{ itemName: 'item2', itemValue: 'item2' },
{ itemName: 'item3', itemValue: 'item3' }
];
viewer.DropDownFieldSettings = {
// Set the name of the form field element.
name: 'DropDown',
// Specify whether the DropDown field is in read-only or read-write mode.
isReadOnly: false,
// Set the visibility of the form field.
visibility: 'visible',

```

```
// Specify whether the field is mandatory or not.
isRequired: false,
// Specify whether to print the DropDown field.
isPrint: true,
// Set the text to be displayed as a tooltip.
tooltip: 'DropDown',
// Set the thickness of the DropDown field. To hide the borders, set the value to 0 (zero).
thickness: 4,
// Set the value of the form field element.
value:'DropDown',
// Set the font family of the DropDown field.
fontFamily: 'Courier',
// Set the font size of the DropDown field.
fontSize: 10,
// Specify the font style
fontStyle: 'None',
// Set the font color of the DropDown field.
color: 'black',
// Set the border color of the DropDown field.
borderColor: 'black',
// Set the background color of the DropDown field.
backgroundColor: '#daeaf7ff',
// Set the alignment of the text.
alignment: 'Left',
// Set the DropDown items.
options: customOptions
}
};
</script>
```



Create form fields with UI interaction

The PDF viewer control provides the option for interaction with Form Fields such as Drag and resize. you can draw a Form Field dynamically by clicking the Form Field icon on the toolbar and draw it in the PDF document. The Form Fields type supported by the PDF Viewer Control are:

- Textbox
- Password
- CheckBox
- RadioButton
- ListBox
- DropDown
- SignatureField
- InitialField

Enable or Disable form designer toolbar

We should inject FormDesigner module and set enableFormDesignerToolbar as true to enable the Form designer icon on the toolbar. By default, enableFormDesignerToolbar is set as true. Use the following code to inject FormDesigner module and to enable the enableFormDesignerToolbar property.

STANDALONE

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").EnableFormDesignerToolbar(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/")).EnableFormDesignerToolbar(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
```
```

Add the form field dynamically

Click the Form Field icon on the toolbar and then click on to the PDF document to draw a Form Field. Refer the below GIF for further reference.

1. Name:
First Name Middle Name Last Name

2. Gender: ☐ Male ☐ Female

3. Date of Birth:
MM DD YY

4. Initial Below:

I DO Agree Initial

I DO NOT Agree Initial

Drag the form field

We provide options to drag the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.

1. Name:
First Name Middle Name Last Name

2. Gender: ☐ Male ☐ Female

3. Date of Birth:
MM DD YY

4. Initial Below:

I DO Agree Initial

I DO NOT Agree Initial

Resize the form field

We provide options to resize the Form Field which is currently selected in the PDF document. Refer the below GIF for further reference.

1 of 1

Submit Form

Add and Edit Form Fields

Workshop

Geek Out Day!

1. Name:

First Name Middle Name Last Name

2. Gender:

☐ Male ☐ Female

3. Date of Birth:

MM DD YY

4. Initial Below:

I DO Agree

I DO NOT Agree

Edit or Update the form field dynamically

The properties of the Form Fields can be edited using the Form Field Properties window. It can be opened by selecting the Properties option in the context menu that appears on the right by clicking the Form Field object. Refer the below image for the properties available to customize the appearance of the Form Field.

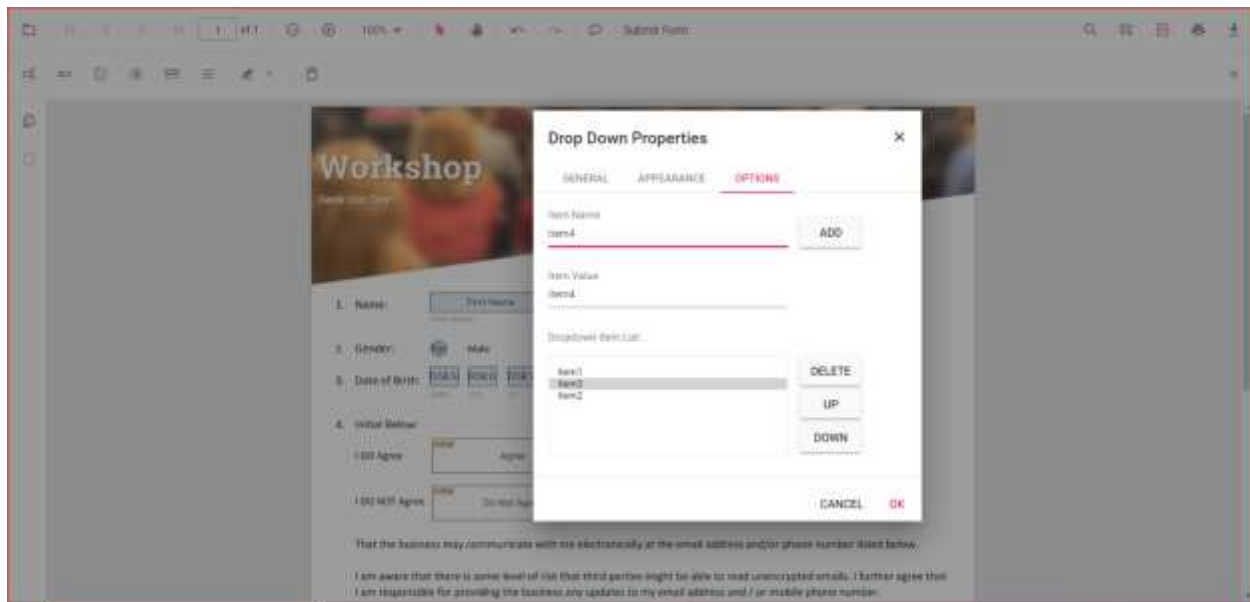
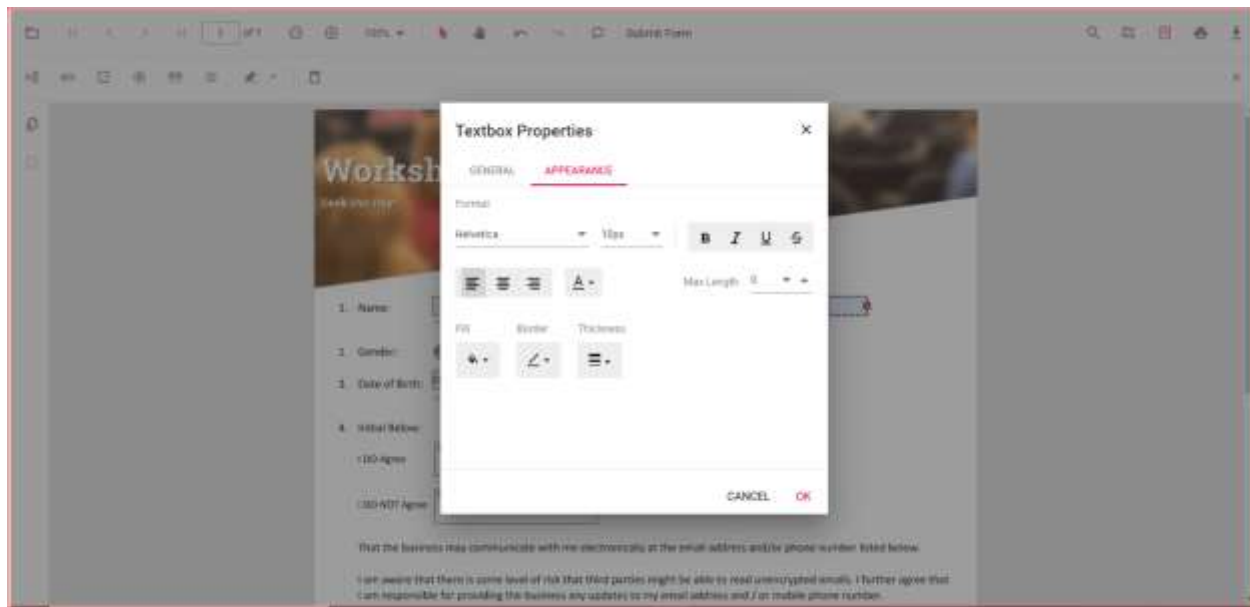
The screenshot shows a web browser window with a 'Textbox Properties' dialog box open. The dialog box has two tabs: 'GENERAL' and 'APPEARANCE'. The 'GENERAL' tab is active, showing the following fields:

- Name:** Surname
- Left Name:** (empty)
- Value:** (empty)
- Form Field Width:** width

At the bottom of the dialog, there are three checkboxes:

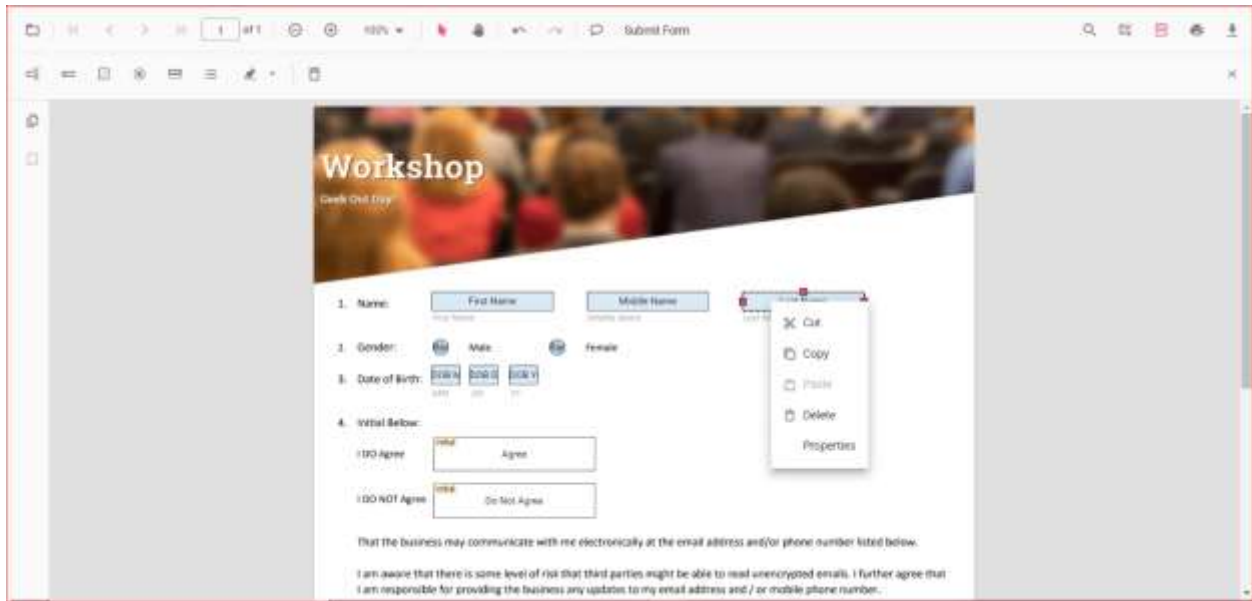
- ☐ Read Only
- ☐ Required
- ☒ Show Printing

The 'OK' button is highlighted in red.



Clipboard operation with form field

The PDF Viewer control supports the clipboard operations such as cut, copy and paste for Form Fields. You can right click on the Form Field object to view the context menu and select to the clipboard options that you would like to perform. Refer the below image for the options in the context menu.



Undo and Redo

We provided support to undo/redo the Form Field actions that are performed at runtime. Use the following code example to perform undo/redo actions.

STANDALONE

```

<<<html
<button id="undo">Undo</button>
<button id="redo">Redo</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
document.getElementById('undo').addEventListener('click', function () {
var pdfviewer = document.getElementById("pdfviewer").ej2_instances[0];
pdfviewer.undo();
});
document.getElementById('redo').addEventListener('click', function () {
var pdfviewer = document.getElementById("pdfviewer").ej2_instances[0];
pdfviewer.redo();
});
</script>
>>>
  
```

SERVER-BACKED

```

<<<html
<button id="undo">Undo</button>
<button id="redo">Redo</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/hi
ve-succinctly.pdf").Render()
</div>
<script>
  
```

```
document.getElementById('undo').addEventListener('click', function () {
    var pdfviewer = document.getElementById("pdfviewer").ej2_instances[0];
    pdfviewer.undo();
});
document.getElementById('redo').addEventListener('click', function () {
    var pdfviewer = document.getElementById("pdfviewer").ej2_instances[0];
    pdfviewer.redo();
});
</script>
```

```

## Print

The PDF Viewer supports printing the loaded PDF file. You can enable/disable the print using the following code snippet.

### STANDALONE

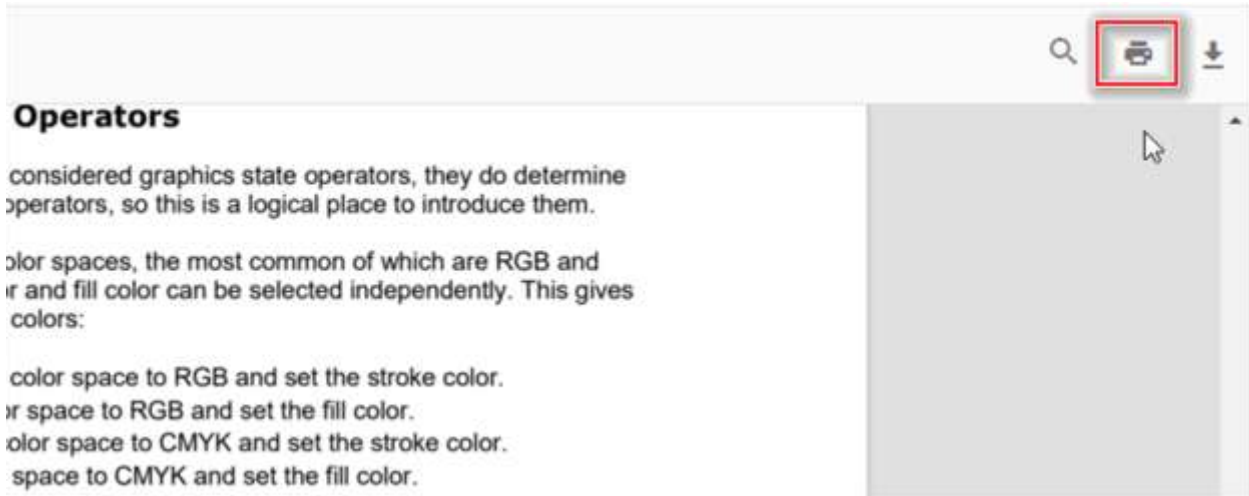
```
```html
<div style="width:100%;height:600px">
    @Html.EJS().PdfViewer("pdfviewer").EnablePrint(true).DocumentLoad("print").DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function print() {
    var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
    pdfViewer.print.print();
}
</script>
```

```

### SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/")).EnablePrint(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function print() {
    var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
    pdfViewer.print.print();
}
</script>
```

```



See also

- [Toolbar items](#)
- [Feature Modules](#)

## Download a PDF document in PDF Viewer component

The PDF Viewer supports downloading the loaded PDF file. You can enable/disable the download using the following code snippet.

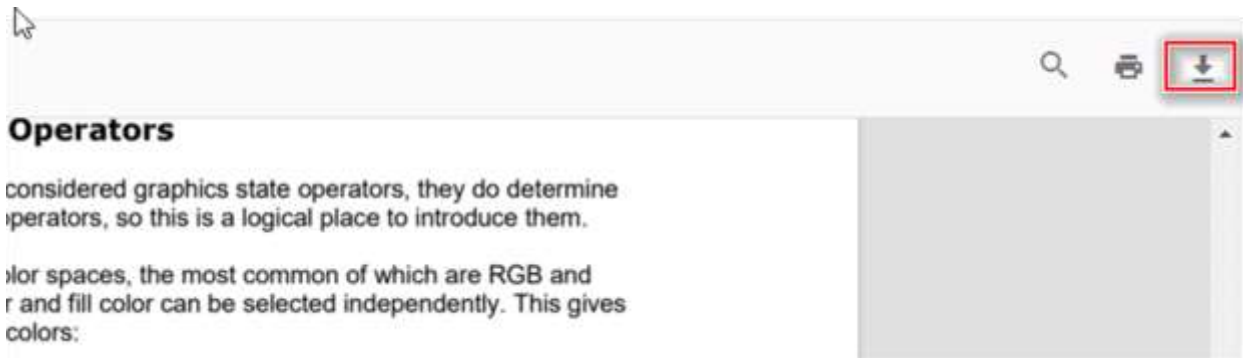
### STANDALONE

```
```html
<div style="width:100%;height:600px">
  @Html.EJS().PdfViewer("pdfviewer").DocumentLoad("download").EnableDownload(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function download() {
  var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
  pdfViewer.download();
}
</script>
```
```

### SERVER-BACKED

```
```html
<div style="width:100%;height:600px">
  @Html.EJS().PdfViewer("pdfviewer").DocumentLoad("download")..ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/")).EnableDownload(true).DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-succinctly.pdf").Render()
</div>
<script>
function download() {
  var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
  pdfViewer.download();
}
```
```

```
</script>
```



See also

- [Toolbar items](#)
- [Feature Modules](#)

## Localization

The text contents provided in the PDF Viewer can be localized using the collection of localized strings for different cultures. By default, the PDF Viewer is localized in “**en-US**”.

The following table shows the default text values used in Syncfusion PDF Viewer in 'en-US' culture:

| Keywords               | Values                                                        |
|------------------------|---------------------------------------------------------------|
| ---                    | ---                                                           |
| PdfViewer              | PDF Viewer                                                    |
| Cancel                 | Cancel                                                        |
| Download file          | Download file                                                 |
| Download               | Download                                                      |
| Enter Password         | This document is password protected. Please enter a password. |
| File Corrupted         | File corrupted                                                |
| File Corrupted Content | The file is corrupted and cannot be opened.                   |
| Fit Page               | Fit page                                                      |
| Fit Width              | Fit width                                                     |
| Automatic              | Automatic                                                     |
| Go To First Page       | Show first page                                               |
| Invalid Password       | Incorrect password. Please try again.                         |
| Next Page              | Show next page                                                |
| OK                     | OK                                                            |
| Open                   | Open file                                                     |

|Page Number|Current page number|  
|Previous Page|Show previous page|  
|Go To Last Page|Show last page|  
|Zoom|Zoom|  
|Zoom In|Zoom in|  
|Zoom Out|Zoom out|  
|Page Thumbnails|Page thumbnails|  
|Bookmarks|Bookmarks|  
|Print|Print file|  
|Password Protected|Password required|  
|Copy|Copy|  
|Text Selection|Text selection tool|  
|Panning|Pan mode|  
|Text Search|Find text|  
|Find in document|Find in document|  
|Match case|Match case|  
|Apply|Apply|  
|GoToPage|Go to page|  
|No matches|Viewer has finished searching the document. No more matches were found|  
|No Text Found|No Text Found|  
|Undo|Undo|  
|Redo|Redo|  
|Annotation|Add or Edit annotations|  
|Highlight|Highlight Text|  
|Underline|Underline Text|  
|Strikethrough|Strikethrough Text|  
|Delete|Delete annotation|  
|Opacity|Opacity|  
|Color edit|Change Color|  
|Opacity edit|Change Opacity|  
|Highlight context|Highlight|  
|Underline context|Underline|  
|Strikethrough context|Strike through|

|Server error|Web-service is not listening. PDF Viewer depends on web-service for all it's features.  
Please start the web service to continue.|

|Open text|Open|

|First text|First Page|

|Previous text|Previous Page|

|Next text|Next Page|

|Last text|Last Page|

|Zoom in text|Zoom In|

|Zoom out text|Zoom Out|

|Selection text|Selection|

|Pan text|Pan|

|Print text|Print|

|Search text|Search|

|Annotation Edit text|Edit Annotation|

The different locale value for the PDF Viewer can be specified using the locale property.

### **STANDALONE**

```

`html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").Locale("ar-
AE").DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
</div>
`

```

### **SERVER-BACKED**

```

`html
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/api/PdfViewer/")).Locale("ar-
AE").DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
</div>
`

```

You have to map the text content based on locale like the following script in sample level.,

```

`html
<script>
ej.base.L10n.load({
'ar-AE': {
'PdfViewer': {

```

'PdfViewer': 'قوات الدفاع الشعبي المشاهد',  
'Cancel': 'إلغاء',  
'Download file': 'تحميل الملف',  
'Download': 'تحميل',  
'Enter Password': 'هذا المستند محمي بكلمة مرور. يرجى إدخال كلمة مرور',  
'File Corrupted': 'ملف تالف',  
'File Corrupted Content': 'الملف تالف ولا يمكن فتحه',  
'Fit Page': 'لائق بدنيا الصفحة',  
'Fit Width': 'لائق بدنيا عرض',  
'Automatic': 'تلقائي',  
'Go To First Page': 'عرض الصفحة الأولى',  
'Invalid Password': 'كلمة سر خاطئة. حاول مرة أخرى',  
'Next Page': 'عرض الصفحة التالية',  
'OK': 'حسنًا',  
'Open': 'افتح الملف',  
'Page Number': 'رقم الصفحة الحالية',  
'Previous Page': 'عرض الصفحة السابقة',  
'Go To Last Page': 'عرض الصفحة الأخيرة',  
'Zoom': 'تكبير',  
'Zoom In': 'تكبير في',  
'Zoom Out': 'تكبير خارج',  
'Page Thumbnails': 'مصغرات الصفحة',  
'Bookmarks': 'المرجعية',  
'Print': 'اطبع الملف',  
'Password Protected': 'كلمة المرور مطلوبة',  
'Copy': 'نسخ',  
'Text Selection': 'أداة اختيار النص',  
'Panning': 'وضع عموم',  
'Text Search': 'بحث عن نص',  
'Find in document': 'ابحث في المستند',  
'Match case': 'حالة مباراة',  
'Apply': 'تطبيق',  
'GoToPage': 'انتقل إلى صفحة'



```
// tslint:disable-next-line:max-line-length
'No matches': 'انتهى العارض من البحث في المستند. لم يتم العثور على مزيد من التطابقات',
'No Text Found': 'لم يتم العثور على نص',
// tslint:disable-next-line:max-line-length
'Undo': 'فك',
'Redo': 'فعل ثانية',
'Annotation': 'إضافة أو تعديل التعليقات التوضيحية',
'Highlight': 'تسليط الضوء على النص',
'Underline': 'تسطير النص',
'Strikethrough': 'نص يتوسطه خط',
>Delete': 'حذف التعليق التوضيحي',
'Opacity': 'غموض',
'Color edit': 'غير اللون',
'Opacity edit': 'تغيير التعقيم',
'Highlight context': 'تسليط الضوء',
'Underline context': 'أكد',
'Strikethrough context': 'يتوسطه',
// tslint:disable-next-line:max-line-length
'Server error': 'خدمة الانترنت لا يستمع. يعتمد قوات الدفاع الشعبي المشاهد على خدمة الويب لجميع ميزاته. يرجى بدء خدمة',
'Open text': 'افتح',
'First text': 'الصفحة الأولى',
'Previous text': 'الصفحة السابقة',
'Next text': 'الصفحة التالية',
'Last text': 'آخر صفحة',
'Zoom in text': 'تكبير',
'Zoom out text': 'تصغير',
'Selection text': 'اختيار',
'Pan text': 'مقالة',
'Print text': 'طباعة',
'Search text': 'بحث',
'Annotation Edit text': 'تحرير التعليق التوضيحي'
}
```

```

}
});
</script>
`

```

### Server Actions with ASP.NET MVC PDFViewer Control

Syncfusion PDF Viewer control is client-server oriented. It processes the PDF document on the server-side and sends the processed PDF data to the client to render the PDF document and for further operations in the PDF Viewer.

The server actions or server methods in the MVC PDF Viewer controller are:

- Load
- RenderPdfPages
- RenderThumbnailImages
- Bookmarks
- RenderAnnotationComments
- Unload
- ExportAnnotations
- ImportAnnotations
- ImportFormFields
- ExportFormFields
- Download
- PrintImages

#### Load action

**Note:** public ActionResult Load(jsonObjects jsonObject)

The [Load](#) action will be triggered initially on loading a PDF file. Syncfusion PDF Viewer control will store the document in the cache based on the hashid during the initial loading of the pdf file. Initially, the request will be sent for 100 pages only and another request will be sent for the remaining pages. So the Load action will get triggered twice when loading a PDF file with pages more than 100, which is called virtual loading.

This action on its first trigger calls the GetDocumentPath method in the PdfViewerController.cs with which the path of the document is determined. The PDF file to be loaded must be located in the folder structure as specified in the GetDocumentPath method.

The Load action calls the [Load](#) method to load the PDF file in the PDF Viewer.

```

`cs
[System.Web.Mvc.HttpPost]
public ActionResult Load(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 MemoryStream stream = new MemoryStream();
 var jsonData = JsonConvert(jsonObject);

```

```
object jsonResult = new object();
if (jsonObject != null && jsonData.ContainsKey("document"))
{
 if (bool.Parse(jsonData["isFileName"]))
 {
 string documentPath = GetDocumentPath(jsonData["document"]);
 if (!string.IsNullOrEmpty(documentPath))
 {
 byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
 stream = new MemoryStream(bytes);
 }
 else
 {
 string fileName = jsonData["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
 if (fileName == "http" || fileName == "https")
 {
 var WebClient = new WebClient();
 byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]);
 stream = new MemoryStream(pdfDoc);
 }
 else
 {
 return this.Content(jsonData["document"] + " is not found");
 }
 }
 }
 else
 {
 byte[] bytes = Convert.FromBase64String(jsonData["document"]);
 stream = new MemoryStream(bytes);
 }
}
jsonResult = pdfviewer.Load(stream, jsonData);
```

```
return Content(JsonConvert.SerializeObject(jsonResult));
}
`cs
```

### RenderPdfPages

**Note:** public ActionResult RenderPdfPages(jsonObjects jsonObject)

Whenever a new page is loaded, [RenderPdfPages](#) action will be called. When a PDF file is loaded with PDF Viewer, only two pages will be loaded initially by hitting this action for loading each page. Further pages will be loaded on demand by hitting this action.

The RenderPdfPages action calls the [GetPage](#) method to render each PDF pages.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult RenderPdfPages(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object jsonResult = pdfviewer.GetPage(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}
`cs
```

### RenderThumbnailImages action

**Note:** public ActionResult RenderThumbnailImages(jsonObjects jsonObject)

The [RenderThumbnailImages](#) action will be triggered initially on loading a PDF file. It renders the thumbnails images of all the pages in the PDF file. When a thumbnail image is clicked, the RenderPdfPages action will be triggered to load the selected page.

This action calls the [GetThumbnailImages](#) method.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult RenderThumbnailImages(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object result = pdfviewer.GetThumbnailImages(jsonData);
 return Content(JsonConvert.SerializeObject(result));
}
`cs
```

### Bookmarks

**Note:** public ActionResult Bookmarks(jsonObjects jsonObject)

The Bookmarks action will be triggered initially on loading a PDF file. All the headings in the PDF file will get added to the Bookmarks collection in the bookmarks panel with the help of this action. When a bookmark is selected, the page containing the selected heading will be rendered or opened.

This action calls the [GetBookmarks](#) method.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult Bookmarks(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object jsonResult = pdfviewer.GetBookmarks(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}
`
```

### RenderAnnotationComments

**Note:** public ActionResult RenderAnnotationComments(jsonObjects jsonObject)

The [RenderAnnotationComments](#) action will be triggered initially on loading a PDF file.

It calls the [GetAnnotationComments](#) method to retrieve the annotation comments in the PDF file.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult RenderAnnotationComments(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object jsonResult = pdfviewer.GetAnnotationComments(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}
`
```

### Unload action

**Note:** public ActionResult Unload(jsonObjects jsonObject)

The [Unload](#) action will be triggered on unloading a PDF file by closing or refreshing the browser.

This action calls the [ClearCache](#) method to clear the cache objects while closing the PDF Viewer.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult Unload(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 pdfviewer.ClearCache(jsonData);
 return this.Content("Document cache is cleared");
}
`
```

#### [ExportAnnotations action](#)

**Note:** public ActionResult ExportAnnotations(jsonObjects jsonObject)

The [ExportAnnotations](#) action will be triggered by clicking either the “Export annotation to JSON file” option or the “Export annotation to XFDF file” option in the annotation toolbar.

This action calls the [ExportAnnotation](#) method to export the annotations in the PDF Viewer to a JSON file or an XFDF file.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult ExportAnnotations(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 string jsonResult = pdfviewer.ExportAnnotation(jsonData);
 return Content((jsonResult));
}
`
```

#### [ImportAnnotations](#)

**Note:** public ActionResult ImportAnnotations(jsonObjects jsonObject)

The [ImportAnnotations](#) action will be triggered by clicking the “Import annotation from XFDF file” option in the annotation toolbar.

This action calls the [ImportAnnotation](#) method to import the annotations from an XFDF file.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult ImportAnnotations(jsonObjects jsonObject)
```

```

{
PdfRenderer pdfviewer = new PdfRenderer();
string jsonResult = string.Empty;
var jsonData = JsonConvert.DeserializeObject(jsonObject);
if (jsonObject != null && jsonData.ContainsKey("fileName"))
{
string documentPath = GetDocumentPath(jsonData["fileName"]);
if (!string.IsNullOrEmpty(documentPath))
{
jsonResult = System.IO.File.ReadAllText(documentPath);
}
else
{
return this.Content(jsonData["document"] + " is not found");
}
}
return Content(JsonConvert.SerializeObject(jsonResult));
}

```

#### ImportFormFields action

**Note:** public ActionResult ImportFormFields(jsonObjects jsonObject)

The [ImportFormFields](#) action will be triggered by calling it from the UI code.

This action calls the [ImportFormFields](#) method to import the form fields into the PDF Viewer from a JSON object or file.

```

`cs
[System.Web.Mvc.HttpPost]
public ActionResult ImportFormFields(jsonObjects jsonObject)
{
PdfRenderer pdfviewer = new PdfRenderer();
var jsonData = JsonConvert.DeserializeObject(jsonObject);
object jsonResult = pdfviewer.ImportFormFields(jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}

```

### ExportFormFields action

**Note:** public ActionResult ExportFormFields(jsonObjects jsonObject)

The [ExportFormFields](#) action will be triggered by clicking the Submit Form option in the PDF Viewer toolbar. The Submit Form option is enabled only when there are form fields. It exports the form fields into a JSON file.

This action calls the [ExportFormFields](#) method to export the form fields into a JSON string.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult ExportFormFields(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 string jsonResult = pdfviewer.ExportFormFields(jsonData);
 return Content(jsonResult);
}
```

### Download action

**Note:** public ActionResult Download(jsonObjects jsonObject)

The [Download](#) action will be triggered by clicking the Download option in the PDF Viewer toolbar to download the PDF file.

This action calls the [GetDocumentAsBase64](#) method to get the PDF file as a base64 string.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult Download(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
 return Content(documentBase);
}
```

### PrintImages

**Note:** public ActionResult PrintImages(jsonObjects jsonObject)

The [PrintImages](#) action will be triggered by clicking the Print option in the PDF Viewer toolbar.



It calls the [GetPrintImage](#) method multiple times to get each page in the PDF file as images to print the entire file.

```
`cs
[System.Web.Mvc.HttpPost]
public ActionResult PrintImages(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object pageImage = pdfviewer.GetPrintImage(jsonData);
 return Content(JsonConvert.SerializeObject(pageImage));
}
`
```

#### Other methods

##### [GetDocumentPath](#)

**Note:** private string GetDocumentPath(string document)

The GetDocumentPath method will be called inside the Load method to get the path of the document to be loaded.

```
`cs
private string GetDocumentPath(string document)
{
 string documentPath = string.Empty;
 if (!System.IO.File.Exists(document))
 {
 var path = HttpContext.Request.PhysicalApplicationPath;
 if (System.IO.File.Exists(path + "/App_Data/" + document))
 documentPath = path + "/App_Data/" + document;
 }
 else
 {
 documentPath = document;
 }
 return documentPath;
}
`
```

*JsonConverter method*

**Note:** public Dictionary<string, string> JsonConverter(jsonObjects results)

The JsonConverter method will be called by other server action methods to convert the jsonObjects into JSON data of type Dictionary<string, string>.

```
`cs
public Dictionary<string, string> JsonConverter(jsonObjects results)
{
 Dictionary<string, object> resultObjects = new Dictionary<string, object>();
 resultObjects = results.GetType().GetProperties(BindingFlags.Instance |
 BindingFlags.Public).ToDictionary(prop => prop.Name, prop => prop.GetValue(results, null));
 var emptyObjects = (from kv in resultObjects
 where kv.Value != null
 select kv).ToDictionary(kv => kv.Key, kv => kv.Value);
 Dictionary<string, string> jsonResult = emptyObjects.ToDictionary(k => k.Key, k => k.Value.ToString());
 return jsonResult;
}
`
```

*GetPlainText method*

**Note:** private HttpResponseMessage GetPlainText(string pageImage)

The GetPlainText method will be useful when you want to perform a server action without downloading the action result in the browser. This method just provides the status of the action result as a HttpResponseMessage by preventing the file download.

```
`cs
private HttpResponseMessage GetPlainText(string pageImage)
{
 var responseText = new HttpResponseMessage(HttpStatusCode.OK);
 responseText.Content = new StringContent(pageImage, System.Text.Encoding.UTF8, "text/plain");
 return responseText;
}
`
```

*jsonObjects class*

The jsonObjects used as the parameter in each method is a class with various properties needed for different features in the PDF Viewer. This class must be defined in the PdfViewerController to perform all the above actions.

```
`cs
public class jsonObjects
```

```
{
public string document { get; set; }
public string password { get; set; }
public string zoomFactor { get; set; }
public string isFileName { get; set; }
public string xCoordinate { get; set; }
public string yCoordinate { get; set; }
public string pageNumber { get; set; }
public string documentId { get; set; }
public string hashId { get; set; }
public string sizeX { get; set; }
public string sizeY { get; set; }
public string startPage { get; set; }
public string endPage { get; set; }
public string stampAnnotations { get; set; }
public string textMarkupAnnotations { get; set; }
public string stickyNotesAnnotation { get; set; }
public string shapeAnnotations { get; set; }
public string measureShapeAnnotations { get; set; }
public string action { get; set; }
public string pageStartIndex { get; set; }
public string pageEndIndex { get; set; }
public string fileName { get; set; }
public string elementId { get; set; }
public string pdfAnnotation { get; set; }
public string importPageList { get; set; }
public string uniqueId { get; set; }
public string data { get; set; }
public string viewPortWidth { get; set; }
public string viewportHeight { get; set; }
public string tilecount { get; set; }
public string freeTextAnnotation { get; set; }
public string signatureData { get; set; }
```

```

public string fieldsData { get; set; }
public string FormDesigner { get; set; }
public string inkSignatureData { get; set; }
public string tileXCount { get; set; }
public string tileYCount { get; set; }
public string digitalSignaturePageList { get; set; }
public string annotationCollection { get; set; }
public string annotationsPageList { get; set; }
public string formFieldsPageList { get; set; }
public string documentLiveCount { get; set; }
public string annotationDataFormat { get; set; }
public bool isCompletePageSizeNotReceived { get; set; }
public bool hideEmptyDigitalSignatureFields { get; set; }
public bool showDigitalSignatureAppearance { get; set; }
public bool digitalSignaturePresent { get; set; }
public bool isAnnotationsExist { get; set; }
public bool isFormFieldAnnotationsExist { get; set; }
}

```

## How To

### Customize the toolbar

The PDF Viewer provides API for user interactions options provided in it's built-in toolbar. Using this we can create our own User Interface for toolbar actions in application level by hiding the default toolbar. The following steps are used to create the custom toolbar for PDF Viewer,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Add EJ2 Toolbar for perform primary actions like Open, Previous page, Next page, Go to page, Print and Download using the following code snippet,

```
`html
```

```
@using Syncfusion.EJ2.Navigations;
```

```
@Html.EJS().Toolbar("topToolbar").Height("56px").Items(new List<ToolbarItem> {
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-open-document-icon", TooltipText =
"Open", Align=ItemAlign.Left, Click="openFile"},
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-previous-page-navigation-
icon", TooltipText = "Previous Page", Align=ItemAlign.Center, Click="previousClicked", Id="previousPage"},

```

```

new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-next-page-navigation-icon", TooltipText =
"Next Page",Align=ItemAlign.Center,Click="nextClicked",Id="nextPage"},

new ToolbarItem {Template="<div class='><input type='text' class='e-input-group e-pv-current-page-
number' id='currentPage'/></div>" ,PrefixIcon = "e-pv-next-page-navigation-icon", TooltipText = "Page
Number",Align=ItemAlign.Center},

new ToolbarItem { Type = ItemType.Input, Template="<div class='><span class='e-pv-total-page-
number' id='totalPage'>of 0</div>" , PrefixIcon = "e-pv-next-page-navigation-icon", TooltipText
= "Page Number",Click="currentPageClicked" ,Align=ItemAlign.Center},

new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-print-document-icon" ,TooltipText =
"Print",Align=ItemAlign.Right,Click="printClicked"},

new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-download-document-icon" ,TooltipText =
"Download",Align=ItemAlign.Right,Click="downloadClicked"},

}).Render()

<input type="file" id="fileUpload" accept=".pdf"
style="display:block;visibility:hidden;width:0;height:0;">

```

**Step 3:** Hide the default toolbar of PDF Viewer using below code snippet,

#### **STANDALONE**

```

`html
@Html.EJS().PdfViewer("pdfviewer")
.EnableToolbar(false).EnableThumbnail(false)
.DocumentLoad("documentLoaded").PageChange("pageChanged")
.DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
`

```

#### **SERVER-BACKED**

```

`html
@Html.EJS().PdfViewer("pdfviewer")
.ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/"))
.EnableToolbar(false).EnableThumbnail(false)
.DocumentLoad("documentLoaded").PageChange("pageChanged")
.DocumentPath("https://cdn.syncfusion.com/content/pdf/hive-
succinctly.pdf").Render()
`

```

**Step 4:** Add EJ2 Toolbar for perform magnification actions in PDF Viewer using following code snippet,

```

`html

@Html.EJS().Toolbar("magnificationToolbar").Items(new List<ToolbarItem> {

new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-fit-page", TooltipText = "Fit to
page",Click="pageFitClicked"},

```

```

new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-zoom-in-icon", TooltipText = "Zoom
in", Click = "zoomInClicked" },
new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-pv-zoom-out-icon", TooltipText = "Zoom
out", Click = "zoomOutClicked" },
}).Render()
,

```

**Step 5:** Add the following style to achieve the custom toolbar styling,

```

`html
<style>
pdfviewer {
display: block;
height: 641px;
}
magnificationToolbar {
background: transparent;
height: auto;
width: 200px;
position: absolute;
z-index: 1001;
top: calc(100% - 110px);
left: calc(100% - 120px);
transform: rotate(90deg);
border-width: 0px;
}
.e-pv-zoom-out-icon {
transform: rotate(-90deg);
}
.material .e-pv-current-page-number {
border-width: 1px;
}
div#magnificationToolbar.e-toolbar .e-toolbar-items {
background: transparent;
}
magnificationToolbar.e-toolbar .e-tbar-btn {
border-radius: 50%;

```

```
min-height: 30px;
min-width: 30px;
}
topToolbar {
top: 0px;
z-index: 1001;
}
.e-pv-current-page-number {
width: 46px;
height: 28px;
text-align: center;
}
.e-icons {
font-family: "e-icons";
font-style: normal;
font-variant: normal;
font-weight: normal;
line-height: 1;
text-transform: none;
}
.e-pv-icon::before {
font-family: 'e-icons';
}
.e-pv-icon-search::before {
font-family: 'e-icons';
font-size: 12px;
}
.e-pv-download-document-icon::before {
content: '\e914';
}
.e-pv-print-document-icon::before {
content: '\e917';
}
```

```
.e-pv-previous-page-navigation-icon::before {
content: '\e910';
}
```

```
.e-pv-next-page-navigation-icon::before {
content: '\e911';
}
```

```
.e-pv-zoom-out-icon::before {
content: '\e912';
}
```

```
.e-pv-zoom-in-icon::before {
content: '\e91d';
}
```

```
.e-pv-fit-page::before {
content: '\e91b';
}
```

```
.e-pv-open-document-icon::before {
content: '\e91c';
}
```

```
@font-face {
font-family: "e-icons";
font-style: normal;
font-weight: normal;
```

```
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgT1MvMj8wSOQAAAEoAAAAVmNtYXDSenLMAAABuAAAAFZnbHlmok0Nt
wAAAjAAAAAPkaGVhZBN3pEcAAADQAAAAANmhoZWEHrwNhAAAArAAAAACRobXR4NsgAAAAAAYAAAAA4b
G9jYQdkBmQAAAIQAAAAHm1heHABHAAwAAABCAAAACBuYW1lD0oZXgAABhQAAALBcG9zdFG4mE4AA
AjYAAAAyAABAAADUv9qAFoEAAAA/+gEAAAABAAAAAAAAAAAAAAAAAADgABAAAAAQAAxsly1F8PPPU
ACwPoAAAAANgsr7EAAAAA2CyvsQAAAAEEAAQAAAAACAACAAAAAAAAAAAEAAAAOACQABAAAAAAAAG
AAAAoACgAAAP8AAAAAAAAAAQPqAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6RDpHQNS/2oAWgQAAJYAAAABAAAAAAAAABAAAAAPoAAD6
AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAA+gAAAPoAAAAAACAAAAWAA
ABQAAwABAAAAFAAEIEIAAAAGAAQAAQAC6RLpHf//AADpEOkU//8AAAAAAAAEABgAKAAAAAQACAAMA
BQAGAAcACAAJAAoACwAMAA0ABAAAAAAAAAAAUACoAZACkAL4A7gEuAVwBcAGEAZ4ByAHyAAAAAQAA
AAD6gMuAAUAAAKBBwkBJwIAAet0/on+iXQDL/4VcwF3/olzAAEAAAAAA+oDLgAFAAATCQEXCQGJAXcB
d3T+Ff4VAY/+iQF3c/4VAesAAAAAAwAAAAEAAQAAAMADwAbAAABITUhBQ4BBY4BJz4BNx4BBRYAFzYA
NyYAJwYAAQACAP4AAoAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAcCAQKPZBATZo6PZBATZo9n+3wY
GASHZ2QEhBgb+3wAAAAADAAAAAAQABAAACwAXACMAAAEjFTMVMzUzNSM1lwEOAQcuASc+ATceAQ
UWABc2ADcmACcGAHAwMCAwMCAACAE2aOj2QQE2aOj2fyEBgEh2dkBIQYG/t/Z2f7fAkCAwMCAwP8A
```



```

o9kEBNmjo9kEBNmj2f7fBgYBI dnZASEGBv7fAAIAAAAAAAwAEAAADAAoAADEhNSEBIQkBIREhAwD9AAEA/
wABgAGA/wD/AIACAP6AAYABgAACAAAAAANABAAADgAaAAABMh4CFREIBRE0Nz4BMycGFREIBRE0JiMh
lgKdCwwHBf7g/uAJBAwKdC8BoAGgX0T+BkQDgAYGCwr9YHZ2AqAOCQQGUS9D/KGrqwNfRIsAAAAACAAA
AAAP/BAAACwAjAAABDgEHLgEnPgE3HgEFHgEXMjY/ARcVATcBlyc3PgE1LgEnDgECgAOQbW2QAwOQbW
2Q/YME2aNGfDIDJAEYf78MyMCKi4E2aOj2QKAbZADA5BtbZADA5Bto9kELioDJDp+/GEBBCQDMnxGo9k
EBNkAAAAQAAAAABAAEAAADAACAFQAZAAABFSE1JRujNSERMxUhNTMRLgEnIQ4BNyE1IQLA/oACQID9A
MACgMABSDf9AddIvwKA/YABwMDAwICA/sDAwAFAN0gBAUmKwAAAAQAAAAACQAQAAAUAAEBENwk
BJwHsU/6HAXpSAmD+YGIbPgE+YgAAAAEAAAAAAkAEAAFAAAARQCQEXCQEBv6HUwHs/hMDnv7C/sJiAa
ABoAABAAAAAAKABAAACwAAERcHFzcXNyc3Jwcn9fVM9PVL9PRL9fQDtfX0TPX1TPT0TPT0AAAAABAAAA
AD8APwAAUACwARABcAAcEzNTM1IQUzFTMRISUHNsm1IwUjFSErIwJ2fvz+hv2K/H7+hgJ2AXr8fv6G/AF6
fvx+fvwBevx+/Px+AXoAAAAAAGAAAAEAAQAAAMAFgAAAREhEScGFREUFhchPgE1ETQmlyEnIQYDgP0AY
h48LQMULtw8Lf5pa/7ULQMA/gACAN8eLf1YLTwDAzwtAigvPYACAAAAAASAN4AAQAAAAAABAAAA
AAQAAAAAQAUAEEAAQAAAAAAGAHABUAAQAAAAAAwAUABwAAQAAAAAABAAUADAAAQAAAAA
ABQALAEQAAQAAAAAABgUAEE8AAQAAAAAAGAsAGMAAAQAAAAACwASAI8AAwABBakAAAAACAEEAA
wABBakAAQAoAKMAAwABBakAAgAOAMsAAwABBakAAwAoANKAAwABBakABAAoAQEAwABBakABQ
AWASKAAwABBakABgAoAT8AAwABBakACgBYAWcAAwABBakACwAkAb8gY3VzdG9tLXRvb2xiYXJbMTkw
OF1SZWd1bGfYy3VzdG9tLXRvb2xiYXJbMTkwOF1jdXN0b20tdG9vbGJhclsxOTA4XVZlcnNpb24gMS4wY3V
zdG9tLXRvb2xiYXJbMTkwOF1Gb250IGdlbmVvYXRIZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3
dy5zeW5jZnVzaW9uLmNvbQAgAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwADgA
XQBSAGUAZwB1AGwAYQByAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwADgAXQB
jAHUAcwB0AG8AbQAtAHQAbwBvAGwAYgBhAHIAWwAxADKAMAA4AF0AVgBIAHIAcwBpAG8AbgAgADE
ALgAwAGMAdQBzAHQAbwBtAC0AdABvAG8AbABiAGEAcgBbADEAOQAwADgAXQBAG8AbgB0ACAAZw
BLAG4AZQByAGEAdABIAGQAIAAB1AHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBIHQQA
cgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAA
gAAAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAE
NAQ4BDwAIVG9wLWlj24LZG93bi1hcnJvdzIKUFZfWm9vbW91dAlQVl9ab29taW4LUFZfRG93bmxvYWQL
UFZfQm9va21hcmsJUFZfU2VhcmNoCFBWX1ByaW50C1BWX1ByZXZpb3VzB1BWX05leHQlUFZfQ2xvc2U
MUFZfRml0VG9QYWdlB1BWX09wZW4AAA==) format('truetype');
}
</style>
`

```

**Note:** The icons are embedded in the font file used in above code snippet.

**Step 6:** Add the following scripts for performing user interaction in PDF Viewer in code behind

```

`html
<script type="text/javascript">
var currentPageBox;
var matchCase = false;
var filename;
window.onload = function () {
currentPageBox = document.getElementById('currentPage');
currentPageBox.value = '1';

```

```
document.getElementById('fileUpload').addEventListener('change', readFile, false);
currentPageBox.addEventListener('keypress', () => {
 var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
 var currentPage = document.getElementById('currentPage');
 if ((event.which < 48 || event.which > 57) && event.which !== 8 && event.which !== 13) {
 event.preventDefault();
 return false;
 }
 else {
 var currentPageNumber = parseInt((currentPage).value);
 if (event.which === 13) {
 if (currentPageNumber > 0 && currentPageNumber <= pdfViewer.pageCount) {
 pdfViewer.navigation.goToPage(currentPageNumber);
 } else {
 (currentPage).value = pdfViewer.currentPageNumber.toString();
 }
 }
 return true;
 }
});
function openFile() {
 document.getElementById('fileUpload').click();
}
function readFile(evt) {
 var uploadedFiles = evt.target.files;
 var uploadedFile = uploadedFiles[0];
 filename = uploadedFiles[0].name;
 var reader = new FileReader();
 reader.readAsDataURL(uploadedFile);
 reader.onload = function () {
 var obj = document.getElementById('pdfviewer').ej2_instances[0];
 var uploadedFileUrl = this.result;
```

```
obj.load(uploadedFileUrl, null);
obj.fileName = filename;
var currentPageBox = document.getElementById('currentPage');
currentPageBox.value = '1';
var pageCount = document.getElementById('totalPage');
pageCount.textContent = 'of ' + obj.pageCount;
}
}
function previousClicked() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.navigation.goToPreviousPage();
}
function nextClicked() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.navigation.goToNextPage();
}
function printClicked() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.print.print();
}
function downloadClicked() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.download();
}
function pageFitClicked() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.magnification.fitToPage();
}
function zoomInClicked() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.magnification.zoomIn();
}
function zoomOutClicked() {
```

```
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
pdfViewer.magnification.zoomOut();
}
function pageChanged() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
var currentPage = document.getElementById('currentPage');
(currentPage).value = pdfViewer.currentPageNumber;
updatePageNavigation();
}
function onCurrentPageBoxKeypress(event) {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var currentPageBox = document.getElementById('currentPage');
if ((event.which < 48 || event.which > 57) && event.which !== 8 && event.which !== 13) {
event.preventDefault();
return false;
}
else {
var currentPageNumber = parseInt(currentPageBox.value);
if (event.which === 13) {
if (currentPageNumber > 0 && currentPageNumber <= viewer.pageCount) {
viewer.navigation.goToPage(currentPageNumber);
}
else {
currentPageBox.value = viewer.currentPageNumber.toString();
}
}
return true;
}
}
function currentPageClicked() {
var currentPage = document.getElementById('currentPage');
(currentPage).select();
}
```

```
function documentLoaded() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
var pageCount = document.getElementById('totalPage');
pageCount.textContent = 'of ' + pdfViewer.pageCount;
updatePageNavigation();
}

function updatePageNavigation() {
var toolbarObj = document.getElementById('topToolbar').ej2_instances[0];
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
if (pdfViewer.currentPageNumber === 1) {
toolbarObj.enableItems(document.getElementById('previousPage'), false);
toolbarObj.enableItems(document.getElementById('nextPage'), true);
} else if (pdfViewer.currentPageNumber === pdfViewer.pageCount) {
toolbarObj.enableItems(document.getElementById('previousPage'), true);
toolbarObj.enableItems(document.getElementById('nextPage'), false);
} else {
toolbarObj.enableItems(document.getElementById('previousPage'), true);
toolbarObj.enableItems(document.getElementById('nextPage'), true);
}
}

function updateZoomButtons() {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
var toolbarObj = document.getElementById('topToolbar').ej2_instances[0];
if (pdfViewer.zoomPercentage <= 50) {
toolbarObj.enableItems(document.getElementById('zoomIn'), true);
toolbarObj.enableItems(document.getElementById('zoomOut'), false);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
} else if (viewer.zoomPercentage >= 400) {
toolbarObj.enableItems(document.getElementById('zoomIn'), false);
toolbarObj.enableItems(document.getElementById('zoomOut'), true);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
} else {
toolbarObj.enableItems(document.getElementById('zoomIn'), true);
```

```

toolbarObj.enableItems(document.getElementById('zoomOut'), true);
toolbarObj.enableItems(document.getElementById('fitPage'), true);
}
}
</script>
`

```

Sample :

<https://ej2.syncfusion.com/aspnetcore/PdfViewer/Default#/material>

### Exporting PDFs as raster images

The PDF Viewer server library allows the PDF document pages to be exported as raster images. Exporting can be done using the `ExportAsImage()` method. This option helps to convert a PDF into an image. The APIs available to export the PDF document pages as images are listed as follows:

#### *ExportAsImage(int pageIndex)*

Exports the specified page as image using the Pdfium rendering engine.

```

`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap bitmapimage = pdfExportImage.ExportAsImage(0);
//Save the exported image in disk
bitmapimage.Save(@"currentDirectory/../../Images/" + "bitmapImage.png");
`

```

#### *ExportAsImage(int pageIndex, float dpiX, float dpiY)*

Exports the specified page as image with respect to the specified DPI values.

```

`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap bitmapimage = pdfExportImage.ExportAsImage(0, 200, 200);
//Save the exported image in disk
bitmapimage.Save(@"currentDirectory/../../Images/" + "bitmapImage.png");
`

```

*ExportAsImage(int pageIndex, SizeF customSize, bool keepAspectRatio)*

Exports the specified page as image with respect to the specified custom size.

```
`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap bitmapimage = pdfExportImage.ExportAsImage(0, new SizeF(200, 300), true);
//Save the exported image in disk
bitmapimage.Save(@"currentDirectory/../../Images/" + "bitmapImage.png");
`
```

*ExportAsImage(int pageIndex, SizeF customSize, float dpiX, float dpiY, bool keepAspectRatio)*

Exports the specified page as image with respect to the custom size and the specified DPI values.

```
`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap bitmapimage = pdfExportImage.ExportAsImage(0, new SizeF(200, 300), 200, 200, true);
//Save the exported image in disk
bitmapimage.Save(@"currentDirectory/../../Images/" + "bitmapImage.png");
`
```

*ExportAsImage(int startIndex, int endIndex)*

Exports the specified pages as images using the Pdfium rendering engine.

```
`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap[] bitmapimage = pdfExportImage.ExportAsImage(0, pdfExportImage.PageCount-1);
`
```

```

for (int i =0; i < pdfExportImage.PageCount; i++)
{
// Save the exported image in disk
bitmapimage[i].Save(@"currentDirectory/../../../../../Images/" + "bitmapImage.png");
}
`

```

*ExportAsImage(int startIndex, int endIndex, float dpiX, float dpiY)*

Exports the specified pages as images with respect to the specified DPI values.

```

`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../../../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap[] bitmapimage = pdfExportImage.ExportAsImage(0, pdfExportImage.PageCount-1, 200, 200);
for (int i =0; i < pdfExportImage.PageCount; i++)
{
//Save the exported image in disk
bitmapimage[i].Save(@"currentDirectory/../../../../../Images/" + "bitmapImage.png");
}
`

```

*ExportAsImage(int startIndex, int endIndex, SizeF customSize, bool keepAspectRatio)*

Exports the specified pages as images with respect to the specified custom size.

```

`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../../../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap[] bitmapimage = pdfExportImage.ExportAsImage(0, pdfExportImage.PageCount-1, new
SizeF(200, 300), false);
for (int i =0; i < pdfExportImage.PageCount; i++)
{
//Save the exported image in disk

```



```

bitmapimage[i].Save(@"currentDirectory/../../Images/" + "bitmapImage.png");
}
`

```

*ExportAsImage(int startIndex, int endIndex, SizeF customSize, float dpiX, float dpiY, bool keepAspectRatio)*

Exports the specified pages as images with respect to the custom size and the specified DPI values.

```

`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExportImage = new PdfRenderer();
//Loads the PDF document
pdfExportImage.Load(@"currentDirectory/../../Data/HTTP Succinctly.pdf");
//Exports the PDF document pages into images
Bitmap[] bitmapimage = pdfExportImage.ExportAsImage(0, pdfExportImage.PageCount-1, new
SizeF(200, 300),200,200,false);
for (int i = 0; i < pdfExportImage.PageCount; i++)
{
//Save the exported image in disk
bitmapimage[i].Save(@"currentDirectory/../../Images/" + "bitmapImage.png");
}
`

```

**Note:** Ensure the provided document path and output image saved locations in your application level.

#### Extract Text from PDF document

The PDF Viewer server library allows you to extract the text from a page along with the bounds. Text extracting can be done using the `ExtractText()` method. Add the following dependency to your application using the [NuGet Package Manager](#).

- Syncfusion.EJ2.PdfViewer.AspNet.Mvc5

**Note:** From Volume 2 2019 release Syncfusion.Pdf.AspNet.Mvc5 and Syncfusion.Compression.Base packages are added as dependency for PDF Viewer control. Ensure the dependency packages are referred in your application properly.

The following code snippet explains how to extract the text from a page.

```

`cs
//Uses the Syncfusion.EJ2.PdfViewer assembly
PdfRenderer pdfExtractText = new PdfRenderer();
pdfExtractText.Load("../Data/HTTP Succinctly.pdf");
//Returns the bounds of the text

```

```
List<Syncfusion.EJ2.PdfViewer.TextData> textCollection = new
List<Syncfusion.EJ2.PdfViewer.TextData>();

//Extracts the text from the first page of the PDF document along with its bounds
string text = pdfExtractText.ExtractText(0, out textCollection);
System.IO.File.WriteAllText("../Data/data.txt", text);
`
```

Sample:

<https://www.syncfusion.com/downloads/support/directtrac/general/ze/ExtractText853154752>

**Note:** Ensure the provided document path and output text saved locations in your application level.

Delete a specific annotation using `deleteAnnotationById`

The PDF Viewer server library allows you to delete a specific annotation from a PDF document. Deleting a specific annotation can be done using the **`deleteAnnotationById()`** method. This method is used to delete a specific annotation using its id.

The following steps are used to delete a specific annotation from PDF Document.

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Use the following code snippet to delete a specific annotation using `deleteAnnotationById()` method.

```
`html
<button type="button" onclick="deleteAnnotationById()">Delete Annotation by Id</button>
<script>
// Delete Annotation by ID.
function deleteAnnotationById() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.annotationModule.deleteAnnotationById(viewer.annotationCollection[0].annotationId);
}
</script>
`
```

Download the sample [how to delete a specific annotation using deleteAnnotationById](#)

Unload the PDF document programmatically

The PDF Viewer library allows you to unload the PDF document being displayed in the PDF Viewer control programmatically using the [unload\(\)](#) method.

The following steps are used to unload the PDF document programmatically.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Add the following code snippet to perform the unload operation.

```
`html
```

```

<button type="button" onclick="unLoad()">Unload Document</button>
<script>
// Unload the PDF document.
function unLoad() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.unload();
}
</script>
`

```

Download the Sample, [how to unload the PDF document programmatically](#)

#### Set author name to annotation

The PDF Viewer server library allows you to set author name to all annotations in the PDF document. Use the **author** property to set author name to all annotations.

The following steps are used to set author name to all annotation for PDF Viewer,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Set author name to annotations using below code snippet.

```

`html
<script>
function documentLoad()
{
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
// set author name to annotation.
viewer.annotationSettings.author = "user1";
}
</script>
`

```

Download the sample [how to set author name to annotation](#)

#### Identify the loaded document is edited

The PDF Viewer server library allows you to identify whether loaded PDF document is edited or not. Use the **isDocumentEdited** property to identify whether the changes made in PDF document.

The following steps are used to identify loaded document is edited in PDF viewer control,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Add the following code snippet with button click event to identity loaded document edited.

```

`html

```

```

<button onclick="beforeEdit()">BeforeDocumentEdit</button>
<button onclick="afterEdit()">AfterDocumentEdit</button>
<script>
function beforeEdit() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
console.log("Is Document Edited = " + viewer.isDocumentEdited);
}
function afterEdit() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
console.log("Is Document Edited = " + viewer.isDocumentEdited);
}
</script>
`

```

Download the sample [how to identify the loaded document is edited](#)

#### Close the comment panel

The PDF Viewer server library allows you to close the comment panel programmatically using the external button event.

The following steps are used to close the comment panel programmatically in PDF Viewer,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Add the following code snippet to close the comment panel using button click event

```

`html
<button type="button" onclick="closeCommentPanel()">CloseCommentPanel</button>
<script>
function closeCommentPanel() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
viewer.viewerBase.navigationPane.closeCommentPanelContainer();
}
</script>
`

```

Download the sample [how to close comment panel](#)

#### Access file name

The PDF Viewer server library allows you to can access the filename of the loaded PDF document using the **documentLoad** and **downloadEnd** event. Using these events, we can access the filename while loading and downloading the PDF document.

The following steps are used to access the file name of loaded PDF document in PDF viewer control,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Access file name using below code snippet,

### **STANDALONE**

```

`html
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/hive-
succinctly.pdf").DocumentLoad("documentLoad").DownloadEnd("documentLoad").Re
nder()
<script>
function documentLoad(args) {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
args.fileName = "pdfsuccinctly.pdf";
//Sets the name of the file to be downloaded
viewer.downloadFileName = "pdfsuccinctly.pdf";
console.log(args);
}
</script>
`

```

### **SERVER-BACKED**

```

`html
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/hive-
succinctly.pdf").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).D
ocumentLoad("documentLoad").DownloadEnd("documentLoad").Render()
<script>
function documentLoad(args) {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
args.fileName = "pdfsuccinctly.pdf";
//Sets the name of the file to be downloaded
viewer.downloadFileName = "pdfsuccinctly.pdf";
console.log(args);
}
</script>
`

```

Download the sample [how to access file name](#)

### **Include Authorization token**

The PDF Viewer server library allows you to include the Authorization token in the PDF Viewer AJAX request using the ajaxRequest headers properties available in AjaxRequestSettings and it will be included in every AJAX request send from PDF Viewer.

The following steps are used to include the Authorization token for PDF viewer control,

**Step 1:** Follow the steps provided in the [link](#) to create simple PDF Viewer sample.

**Step 2:** Add the following code snippet to include the authorization token,

```

`html

@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).Docume
ntPath("PDF_Succinctly.pdf").AjaxRequestSettings(new PdfViewerAjaxRequestSettings { WithCredentials

```

```
= true, AjaxHeaders = new object[] { new { headerName = "Testingabc", headerValue = "Testing123" } }
}).Render()
```

Download the sample [how to include authorization token](#)

#### Achieve Load balancing environment in MVC framework

The PDF Viewer server library allows you to achieve a Load balancing environment in the MVC framework using the .NET 4.5 framework with the help of **StackExchange Redis package V1.2.6**.

**Note:** We have considered the support from the .NET framework 4.5 version and it won't work in the lower .NET framework version.

#### *Steps to achieve load balancing environment in the MVC framework*

**Step 1:** Install the [StackExchange.Redis.StrongName](#) NuGet in the web-service sample.

**Step 2:** Create a new class in the web-service project (Example: CacheManager.cs) and implement the **Syncfusion.EJ2.PdfViewer.ICacheManager** interface in the class.

```
`cs
public class CacheManager:ICacheManager{}
```

**Step 3:** Connect the Redis database in the constructor by using the **redisconnectionstring** as provided in the below code.

```
`cs
private IDatabase m_redisDatabase = null;
private int m_slidingExpiration = 0;
public CacheManager(string redisConnectionString = null, int slidingExpiration = 0)
{
 if (!string.IsNullOrEmpty(redisConnectionString))
 {
 ConnectionMultiplexer connection = new Lazy<ConnectionMultiplexer>(() =>
 {
 return ConnectionMultiplexer.Connect(redisConnectionString);
 }).Value;
 if (connection != null)
 {
 m_redisDatabase = connection.GetDatabase();
 m_slidingExpiration = slidingExpiration;
 }
 }
}
```

```
}
,
```

**Step 4:** Override the **ICacheManager** interface methods into the newly created class for adding, retrieving, and deleting the data from the Redis cache.

```
`cs
//Adding the data in the redis database.
public void AddCache(string key, byte[] data)
{
 if (m_redisDatabase != null)
 {
 TimeSpan time = mslidingExpiration > 0 ? new TimeSpan(0, mslidingExpiration, 0) : new TimeSpan(24, 0, 0); // Provided the sliding expiration time
 m_redisDatabase.StringSet(key, data, time);
 }
}

//Retrieve the data in the redis database.
public byte[] GetCache(string key)
{
 byte[] documentContent = null;
 if (m_redisDatabase != null)
 {
 documentContent = m_redisDatabase.StringGet(key);
 }
 return documentContent;
}

//Delete the data in the redis database.
public void DeleteCache(string key)
{
 if (m_redisDatabase != null)
 {
 mredisDatabase.KeyDelete("PDFCONTENT" + key);
 mredisDatabase.KeyDelete("PDFINFO" + key);
 }
}
```

**Step 5:** In the controller, create a new object by passing the **redisconnectionstring** and the sliding expiration time.

```
`cs
```

```
pdfviewer.CacheManager = new CacheManager(redisconnectonstring, slidingexpiration);
```

**Note:** The sliding expiration is the time in which the data has to be stored in the cache for a specific minutes. If 0 then it will store for 24 hours.

**Step 6:** Set the newly created object to **CacheManager** property by using the below code-snippet.

```
`cs
```

```
PdfRenderer pdfviewer = new PdfRenderer();
```

```
pdfviewer.CacheManager = new CacheManager(redisconnectonstring, 0);
```

**Step 7:** Use this code in all our controller methods (Load, RenderPdfPage).

```
`cs
```

```
public string redisconnectonstring = "xxxx,ssl=True,abortConnect=False,syncTimeout=100000";
```

```
[System.Web.Mvc.HttpPost]
```

```
public object Load(Dictionary<string, string> jsonObject)
```

```
{
```

```
PdfRenderer pdfviewer = new PdfRenderer();
```

```
MemoryStream stream = new MemoryStream();
```

```
object jsonResult = new object();
```

```
if (jsonObject != null && jsonObject.ContainsKey("document"))
```

```
{
```

```
if (bool.Parse(jsonObject["isFileName"]))
```

```
{
```

```
string documentPath = GetDocumentPath(jsonObject["document"]);
```

```
if (!string.IsNullOrEmpty(documentPath))
```

```
{
```

```
byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
```

```
stream = new MemoryStream(bytes);
```

```
}
```

```
else
```

```
{
```



```

return (jsonObject["document"] + " is not found");
}
}
else
{
byte[] bytes = Convert.FromBase64String(jsonObject["document"]);
stream = new MemoryStream(bytes);
}
}

pdfviewer.CacheManager = new CacheManager(redisconnectonstring, 0);
jsonResult = pdfviewer.Load(stream, jsonObject);
return (JsonConvert.SerializeObject(jsonResult));
}

```

**Note: Add the bold lines in all the controller methods.**

Find the Web-service sample and Nuget package below:

- [Web service](#)
- [Nuget](#)

#### Extract Text using TextLineCollection

The PDF Viewer server library allows you to extract the text from a page along with the bounds using TextLineCollection. Text extracting can be done using the **ExtractText()** method.

Add the following dependency to your application using the **NuGet Package Manager**.

- Syncfusion.EJ2.PdfViewer.AspNet.Mvc5

**Note:** From Volume 2 2019 release Syncfusion.Pdf.AspNet.Mvc5 and Syncfusion.Compression.Base packages are added as dependency for PDF Viewer control. Ensure the dependency packages are referred in your application properly.

The following code snippet explains how to extract the text from a page using TextLineCollection.

```

`cs
var path = @"currentDirectory/../../Data/Simple.pdf";
var fileInfo = new FileInfo(path);
var docStream = new FileStream(fileInfo.FullName, FileMode.Open, FileAccess.Read);
// Load the PDF document.
PdfLoadedDocument document = new PdfLoadedDocument(docStream);

```

```
// Loading page collections
PdfPageBase page = document.Pages[0] as PdfLoadedPage;
//Extract text from the page.
var text = page.ExtractText(out TextLineCollection textLineCollection);
`
```

Find the sample [How to extract text using TextLineCollection](#)

**Note:** Ensure the provided document path and output text saved locations in your application level.

#### Load the PDF document

The PDF Viewer server library allows you to load the PDF document in the PDF Viewer Control. PDF Document can be loaded by adding the necessary JSON object Properties in the **PdfViewerController.cs** file.

The following steps are used to load a PDF document.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Add the following code snippet in PdfViewerController.cs file to load a PDF document.

```
`cs
public class jsonObjects
{
 public string document { get; set; }
 public string password { get; set; }
 public string zoomFactor { get; set; }
 public string isFileName { get; set; }
 public string xCoordinate { get; set; }
 public string yCoordinate { get; set; }
 public string pageNumber { get; set; }
 public string documentId { get; set; }
 public string hashId { get; set; }
 public string sizeX { get; set; }
 public string sizeY { get; set; }
 public string startPage { get; set; }
 public string endPage { get; set; }
 public string stampAnnotations { get; set; }
 public string textMarkupAnnotations { get; set; }
 public string stickyNotesAnnotation { get; set; }
 public string shapeAnnotations { get; set; }
}
```

```

public string measureShapeAnnotations { get; set; }
public string action { get; set; }
public string pageStartIndex { get; set; }
public string pageEndIndex { get; set; }
public string fileName { get; set; }
public string elementId { get; set; }
public string pdfAnnotation { get; set; }
public string importPageList { get; set; }
public string uniqueId { get; set; }
public string data { get; set; }
public string viewPortWidth { get; set; }
public string viewportHeight { get; set; }
public string tilecount { get; set; }
public string isCompletePageSizeNotReceived { get; set; }
public string freeTextAnnotation { get; set; }
public string signatureData { get; set; }
public string fieldsData { get; set; }
}
`

```

Find the sample [how to load the PDF Document in MVC PDF Viewer](#)

Show the notification dialog in UI When form fields are empty

The PDF Viewer server library allows you to show the notification dialog in UI when fields in the form are not filled or empty using the following properties and events below,

- **EnableFormFieldsValidation**
- **ShowNotificationDialog**
- **validateFormFields**

The following steps are used to show the notification dialog in UI.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Use the following code snippet to show the notification dialog when form fields are empty.

```
`cs
```

```

@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/PdfViewer/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/form-filling-document.pdf").ValidateFormFields("validateFormFields").EnableFormFieldsValidation(true).ShowNotificationDialog(false).Render()

```

```
<script>
function validateFormFields(args) {
var nonfilledFormFields = args.nonFillableFields;
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var errorMessage = "Kindly fill all the form fieds";
viewer.showNotificationPopup(errorMessage);
}
</script>
`
```

Find the sample of [how to show the notification dialog in UI When form fields are empty](#)

#### Load PDF documents dynamically

The PDF Viewer server library allows to switch or load the PDF documents dynamically after the initial load operation. To achieve this, load the PDF document as a base64 string or file name in PDF Viewer control using the **Load()** method dynamically.

The following steps are used to load the PDF document dynamically.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Use the following code snippet to load PDF document using base64 string.

```
`html
<button type="button" onclick="load1()">LoadDocumentFromBase64</button>
<script>
// Load a Base64 String
function load1() {
// Sending Ajax request to the controller to get base 64 string
$.ajax({
url: '/PdfViewer/GetDocument',
type: 'POST',
cache: false,
processData: false,
contentType: false,
success: function (data) {
debugger;
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.load(data, null);
},
```

```

error: function (msg, textStatus, errorThrown) {
 debugger;
 alert('Exception' + msg.responseText);
}
});
}
</script>
`

```

**Step 3:** Add the following code snippet in PdfViewerController.cs file to get the base64 string of the given document.

```

`cs
public ActionResult GetDocument()
{
 var docBytes = System.IO.File.ReadAllBytes(GetDocumentPath("PDF_Succinctly.pdf"));
 string docBase64 = "data:application/pdf;base64," + Convert.ToBase64String(docBytes);
 return Content(docBase64);
}
`

```

**Step 4:** Use the following code snippet to load PDF document using document name.

```

`html
<button type="button" onclick="load2()">LoadDocumentFromBase64</button>
<script>
// load document using document name.
function load2() {
 var viewer = document.getElementById('pdfViewer').ej2_instances[0];
 viewer.load("HTTP Succinctly.pdf", null)
}
</script>
`

```

**Tips:** Also can add the base64 string directly in the DocumentPath() method.

Download the sample, [how to load PDF documents dynamically](#)

#### Import and Export annotation as object

The PDF Viewer library allows you to import annotations from objects or streams instead of loading it as a file. To import such annotation objects, the PDF Viewer control must export the PDF annotations as

objects using the [ExportAnnotationsAsObject\(\)](#) method. Only the annotations objects that are exported from the PDF Viewer can be imported.

The following steps are used to import and export annotation as object.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Use the following code snippet to perform import and export annotation.

```
`html
<button type="button" onclick="exportAnnotation()">Export Annoatation</button>
<button type="button" onclick="importAnnotation()">Import Annoatation</button>
<script>
var exportObject;
//Export annotation as object.
function exportAnnotation() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.exportAnnotationsAsObject().then(function (value) {
exportObject = value
});
}
//Import annotation that are exported as object.
function importAnnotation() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
viewer.importAnnotation(JSON.parse(exportObject));
}
</script>
`
```

Find the Sample, [how to import and export annotation as object](#)

[Open Thumbnail pane programmatically](#)

The PDF Viewer library allows you to open the thumbnail pane programmatically using the [openThumbnailPane\(\)](#) method.

The following steps are used to open the thumbnail.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Use the following code snippet to open thumbnail.

```
`html
<button type="button" onclick="openThumbnail()">Open Thumbnail Pane</button>
<script>
```

```
function openThumbnail() {
var viewer = document.getElementById('pdfViewer').ej2_instances[0];
// Open Thumbnail Pane.
viewer.thumbnailViewModule.openThumbnailPane();
}
</script>
`
```

Find the sample, [how to open the thumbnail pane programmatically](#)

#### Redirect to a home page after submitting PDF forms

The PDF Viewer library allows you to redirect to a home page after submitting PDF forms in the PDF Viewer control programmatically using the [ExportSuccess](#) event. This event triggers once the form fields exporting is succeeded in the PDF Viewer.

The following example steps are used to redirect to a home page programmatically after submitting PDF forms.

**Step 1:** Follow the steps provided in the [link](#) to create a simple PDF Viewer sample.

**Step 2:** Add the following codes in new View pages HomePage.cshtml and FormFillingData.cshtml inside a folder(say FormFilling).

FormFillingData.cshtml:

```
`html
@{
ViewBag.Title = "FormFillingData";
}
@using Syncfusion.EJ2
@using Syncfusion.EJ2.PdfViewer
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/FormFilling/")).DocumentPath("FormFillingDocument.pdf").ToolBarSettings(new
Syncfusion.EJ2.PdfViewer.PdfViewerToolBarSettings { ShowTooltip = true, ToolbarItems = "SubmitForm,
DownloadOption, PrintOption"
}).EnableFormFields(true).ValidateFormFields("validateFormFields").EnableFormFieldsValidation(true).ShowNotificationDialog(false).ExportSuccess("exportSuccess").Render()
<style>
pdfviewer {
display: block;
}
</style>
<script>
```

```
function exportSuccess(args) {
//call the home page action
window.open(window.location.href + "FormFilling/HomePage", "_self")
}
function validateFormFields(args) {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var errorMessage = "Required Field(s): ";
var forms = viewer.formFieldCollections;
var flag = false;
var radioGroupName = "";
for (var i = 0; i < forms.length; i++) {
var text = "";
if (forms[i].isRequired == true) {
if (forms[i].type.toString() == "Checkbox" && forms[i].isChecked == false) {
text = forms[i].name;
}
else if (forms[i].type == "RadioButton" && flag == false) {
radioGroupName = forms[i].name;
if (forms[i].isSelected == true)
flag = true;
}
else if (forms[i].type.toString() != "Checkbox" && forms[i].type != "RadioButton" && forms[i].value ==
"") {
text = forms[i].name;
}
if (text != "") {
if (errorMessage == "Required Field(s): ") {
errorMessage += text;
}
else {
errorMessage += ", " + text;
}
}
}
}
```



```

}
}
if (!flag && radioGroupName != "") {
if (errorMessage == "Required Field(s): ")
errorMessage += radioGroupName;
else
errorMessage += ", " + radioGroupName;
}
if (errorMessage != "Required Field(s): ") {
viewer.showNotificationPopup(errorMessage);
}
}
</script>
`

```

HomePage.cshtml:

```

`html
@{
ViewBag.Title = "HomePage";
}
<h2>HomePage</h2>
`

```

**Step 3:** Create a new Controller file called FormFillingController.cs inside the Controllers and add the below code to it.

FormFillingController.cs:

```

`cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Syncfusion.EJ2.PdfViewer;
using Newtonsoft.Json;
using System.IO;

```

```
using System.Reflection;
using Syncfusion.Pdf.Parsing;
using System.Net.Http;
using System.Net;
using System.Text.RegularExpressions;
using Syncfusion.Pdf;
using Syncfusion.Pdf.Graphics;
using System.Drawing;
using Syncfusion.Pdf.Interactive;
using System.Web.Http;
using System.Threading.Tasks;
namespace EJ2PdfViewer.Controllers
{
 public class FormFillingController : Controller
 {
 // GET: FormFilling
 public ActionResult FormFillingData()
 {
 return View();
 }
 public ActionResult HomePage()
 {
 return View("HomePage");
 }
 [System.Web.Mvc.HttpPost]
 public ActionResult Load(jsonObjects jsonObject)
 {
 PdfRenderer pdfviewer = new PdfRenderer();
 PdfRenderer.ReferencePath = "C:/"; //_hostingEnvironment.WebRootPath + "/";
 MemoryStream stream = new MemoryStream();
 var jsonData = JsonConverter(jsonObject);
 object jsonResult = new object();
 if (jsonObject != null && jsonData.ContainsKey("document"))
```

```
{
if (bool.Parse(jsonData["isFileName"]))
{
string documentPath = GetDocumentPath(jsonData["document"]);
if (!string.IsNullOrEmpty(documentPath))
{
byte[] bytes = System.IO.File.ReadAllBytes(documentPath);
stream = new MemoryStream(bytes);
}
else
{
string fileName = jsonData["document"].Split(new string[] { "://" }, StringSplitOptions.None)[0];
if (fileName == "http" || fileName == "https")
{
var WebClient = new WebClient();
byte[] pdfDoc = WebClient.DownloadData(jsonData["document"]);
stream = new MemoryStream(pdfDoc);
}
else
{
return this.Content(jsonData["document"] + " is not found");
}
}
}
else
{
byte[] bytes = Convert.FromBase64String(jsonData["document"]);
stream = new MemoryStream(bytes);
}
}

PdfLoadedDocument document = new PdfLoadedDocument(stream);
PdfBookmark bookmark = document.Bookmarks.Add("Page 1");
bookmark.Destination = new PdfDestination(document.Pages[0]);
```

```
bookmark.Color = Color.Red;
bookmark.TextStyle = PdfTextStyle.Bold;
bookmark.Destination.Location = new PointF(20, 20);
MemoryStream docStream = new MemoryStream();
document.Save(docStream);
docStream.Position = 0;
document.Close(true);
jsonResult = pdfviewer.Load(docStream, jsonData);
return Content(JsonConvert.SerializeObject(jsonResult));
}

public ActionResult GetDocument()
{
 var docBytes = System.IO.File.ReadAllBytes(GetDocumentPath("PDF_Succinctly.pdf"));
 string docBase64 = "data:application/pdf;base64," + Convert.ToBase64String(docBytes);
 return Content(docBase64);
}

public ActionResult SaveDocument(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConvert.SerializeObject(jsonObject);
 string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
 string base64String = documentBase.Split(new string[] { "data:application/pdf;base64," },
 StringSplitOptions.None)[1];
 if (base64String != null || base64String != string.Empty)
 {
 byte[] byteArray = Convert.FromBase64String(base64String);
 var path = GetDocumentPath(jsonData["documentId"]);
 // Document is saved in the project location
 System.IO.File.WriteAllBytes(path, byteArray);
 }
 return Content(string.Empty);
}

public Dictionary<string, string> JsonConvert(jsonObjects results)
```

```
{
Dictionary<string, object> resultObjects = new Dictionary<string, object>();
resultObjects = results.GetType().GetProperties(BindingFlags.Instance | BindingFlags.Public)
.ToDictionary(prop => prop.Name, prop => prop.GetValue(results, null));
var emptyObjects = (from kv in resultObjects
where kv.Value != null
select kv).ToDictionary(kv => kv.Key, kv => kv.Value);
Dictionary<string, string> jsonResult = emptyObjects.ToDictionary(k => k.Key, k => k.Value.ToString());
return jsonResult;
}

[System.Web.Mvc.HttpPost]
public ActionResult ExportAnnotations(jsonObjects jsonObject)
{
PdfRenderer pdfviewer = new PdfRenderer();
var jsonData = JsonConverter(jsonObject);
string jsonResult = pdfviewer.ExportAnnotation(jsonData);
return Content((jsonResult));
}

[System.Web.Mvc.HttpPost]
public ActionResult ImportAnnotations(jsonObjects jsonObject)
{
PdfRenderer pdfviewer = new PdfRenderer();
string jsonResult = string.Empty;
var jsonData = JsonConverter(jsonObject);
if (jsonObject != null && jsonData.ContainsKey("fileName"))
{
string documentPath = GetDocumentPath(jsonData["fileName"]);
if (!string.IsNullOrEmpty(documentPath))
{
jsonResult = System.IO.File.ReadAllText(documentPath);
}
}
else
{

```

```
return this.Content(jsonData["document"] + " is not found");
}
}
return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult ImportFormFields(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object jsonResult = pdfviewer.ImportFormFields(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult ExportFormFields(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 string jsonResult = pdfviewer.ExportFormFields(jsonData);
 return Content(jsonResult);
}
[System.Web.Mvc.HttpPost]
public ActionResult RenderPdfPages(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object jsonResult = pdfviewer.GetPage(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}
[System.Web.Mvc.HttpPost]
public ActionResult Unload(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
```

```
var jsonData = JsonConvert(jsonObject);
pdfviewer.ClearCache(jsonData);
return this.Content("Document cache is cleared");
}

[System.Web.Mvc.HttpPost]
public ActionResult RenderThumbnaiImages(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConvert(jsonObject);
 object result = pdfviewer.GetThumbnaiImages(jsonData);
 return Content(JsonConvert.SerializeObject(result));
}

[System.Web.Mvc.HttpPost]
public ActionResult Bookmarks(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConvert(jsonObject);
 object jsonResult = pdfviewer.GetBookmarks(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}

[System.Web.Mvc.HttpPost]
public ActionResult RenderAnnotationComments(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConvert(jsonObject);
 object jsonResult = pdfviewer.GetAnnotationComments(jsonData);
 return Content(JsonConvert.SerializeObject(jsonResult));
}

[System.Web.Mvc.HttpPost]
public ActionResult Download(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConvert(jsonObject);
```

```
string documentBase = pdfviewer.GetDocumentAsBase64(jsonData);
return Content(documentBase);
}
[System.Web.Mvc.HttpPost]
public ActionResult PrintImages(jsonObjects jsonObject)
{
 PdfRenderer pdfviewer = new PdfRenderer();
 var jsonData = JsonConverter(jsonObject);
 object pageImage = pdfviewer.GetPrintImage(jsonData);
 return Content(JsonConvert.SerializeObject(pageImage));
}
private HttpResponseMessage GetPlainText(string pageImage)
{
 var responseText = new HttpResponseMessage(HttpStatusCode.OK);
 responseText.Content = new StringContent(pageImage, System.Text.Encoding.UTF8, "text/plain");
 return responseText;
}
private string GetDocumentPath(string document)
{
 string documentPath = string.Empty;
 if (!System.IO.File.Exists(document))
 {
 var path = HttpContext.Request.PhysicalApplicationPath;
 if (System.IO.File.Exists(path + "/App_Data/" + document))
 documentPath = path + "/App_Data/" + document;
 }
 else
 {
 documentPath = document;
 }
 return documentPath;
}
}
```



```
public class jsonObjects
{
 public string document { get; set; }
 public string password { get; set; }
 public string zoomFactor { get; set; }
 public string isFileName { get; set; }
 public string xCoordinate { get; set; }
 public string yCoordinate { get; set; }
 public string pageNumber { get; set; }
 public string documentId { get; set; }
 public string hashId { get; set; }
 public string sizeX { get; set; }
 public string sizeY { get; set; }
 public string startPage { get; set; }
 public string endPage { get; set; }
 public string stampAnnotations { get; set; }
 public string textMarkupAnnotations { get; set; }
 public string stickyNotesAnnotation { get; set; }
 public string shapeAnnotations { get; set; }
 public string measureShapeAnnotations { get; set; }
 public string action { get; set; }
 public string pageStartIndex { get; set; }
 public string pageEndIndex { get; set; }
 public string fileName { get; set; }
 public string elementId { get; set; }
 public string pdfAnnotation { get; set; }
 public string importPageList { get; set; }
 public string uniqueId { get; set; }
 public string data { get; set; }
 public string viewPortWidth { get; set; }
 public string viewportHeight { get; set; }
 public string tilecount { get; set; }
 public string isCompletePageSizeNotReceived { get; set; }
```

```

public string freeTextAnnotation { get; set; }
public string signatureData { get; set; }
public string fieldsData { get; set; }
public string formDesigner { get; set; }
public string inkSignatureData { get; set; }
}
}
`

```

**Step 4:** Modify the RegisterRoutes method in the RouteConfig.cs file inside the App\_Start folder to map the default path.

```

`cs
public static void RegisterRoutes(RouteCollection routes)
{
 routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
 routes.MapRoute(
 name: "Default",
 url: "{controller}/{action}/{id}",
 defaults: new { controller = "FormFilling", action = "FormFillingData", id = UrlParameter.Optional }
);
}
`

```

Now, run the sample, fill in the form fields, and click the Submit Form option. You will be redirected to the home page.

Download the sample, [how to redirect to a home page programmatically after submitting PDF forms](#).

#### Resolve the Pdfium issue

The issue, “The type initializer for 'Syncfusion.EJ2.PdfViewer.PdfiumNative' threw an exception” is due to the write access permission denied in the environment. The pdfium.dll will be created based on the operating system at the runtime. Due to the denial of permission, the pdfium.dll file couldn't have been created. So, copy the below x64 and x86 folders and paste them inside the folder into your project to resolve the issue or enable the write permission for that folder.

Pdfium dll: [pdfium.dll](#)

**Note:** Use both the client and server-side of the same version in your project.

Ensure whether the pdfium.dll file is created in your project during runtime. Else, place the pdfium assemblies in any of the production environment locations and refer to the path by using the ReferencePath API.

For example, if the Pdfium assembly is available in this path C:\Pdfium\x64 or D:\Pdfium\x86, the reference path should be PdfRenderer.ReferencePath = "C:/";

The parent folder has to be provided as the path in the ReferencePath API.

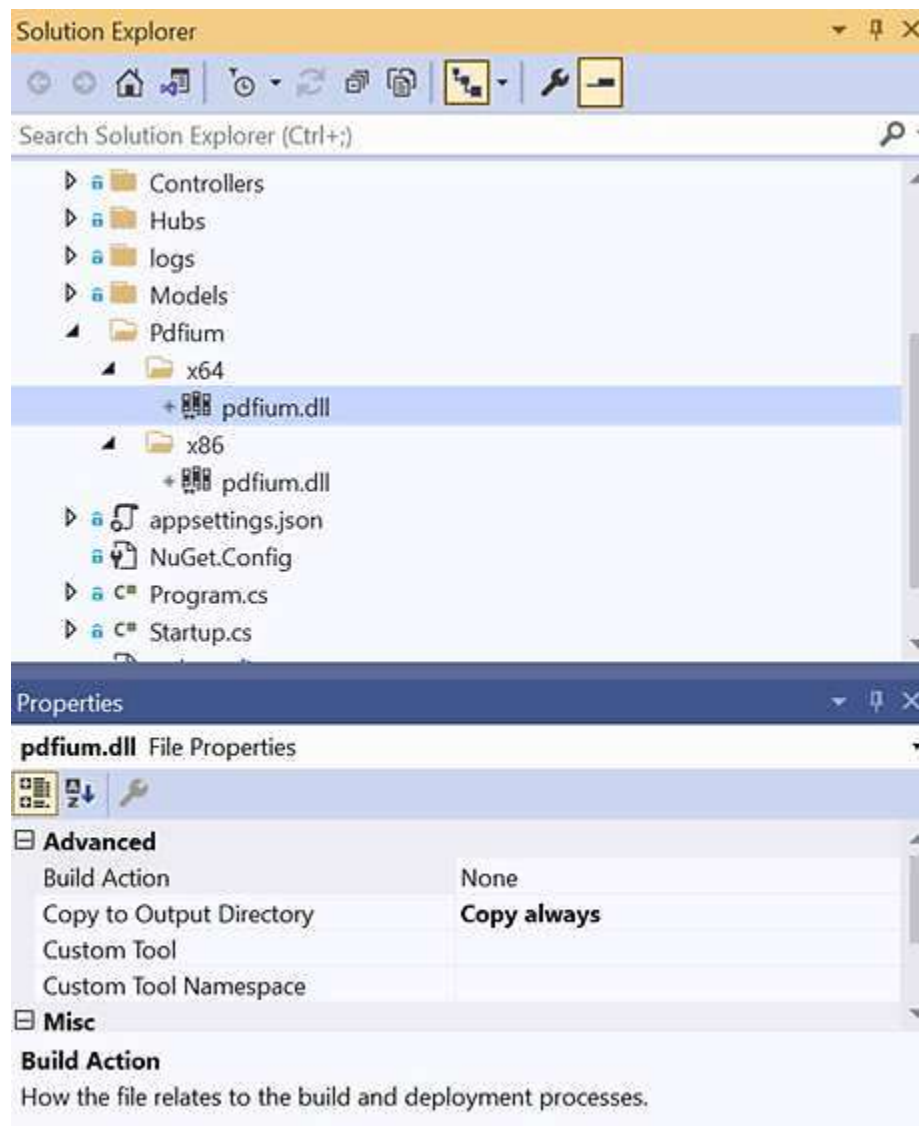
**Note:** Provide this path in the Load method of the PDFViewController.cs

**Note:** PdfRenderer PdfRenderer = new PdfRenderer();

<br/> PdfRenderer.ReferencePath = @"C:/";

*Steps to refer the Pdfium.dll*

1. Extract the given file (Pdfium folder) and copy it in the sample's root directory (parallel to Controllers folder)
2. Right-click on the pdfium.dll (both the X64 and X86 folder) and then choose the "Copy to Output Directory" property from the property window and set its value to "copy always". This setting will ship the pdfium.dll assembly to the published location.



3. Provide this path in the Load method of the PDFViewerController.cs

**Note:** You need to refer the ParentFolder up to the x64/x86 folder.

```
PdfRenderer.ReferencePath = _hostingEnvironment.ContentRootPath + /Pdfium/;
```

4. Build and publish the application.

Also, install only the package related to that OS, then build and run the project on that platform. For Windows, Linux, and OSX operating systems, use the following corresponding libraries:

- Syncfusion.EJ2.PdfViewer.AspNet.Core.Linux
- Syncfusion.EJ2.PdfViewer.AspNet.Core.Windows
- Syncfusion.EJ2.PdfViewer.AspNet.Core.OSX

Following these steps should resolve the issue.

#### How to add the date to the signature text

To add a date with the signature text in Syncfusion PDF Viewer, use the `updateFormFieldsValue()` method. Add a signature field to a PDF document and get the signature field in a PDF using the `retrieveFormFields()` method and store it in a variable. Modify the value of the signature field and get the current date. Use the `Date()` method to get the current date and format it as desired.

Update the signature field in a PDF using the `updateFormFieldsValue()` method and pass the modified signature field as a parameter.

#### **STANDALONE**

```

` ``html
<div>
<div style="width:100%;height:600px">
<button type="button" onclick="signatureWithDate()">Add Signature with the
time</button>
@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/form-filling-document.pdf").Render()
</div>
</div>
<script>
function signatureWithDate() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formFields = viewer.retrieveFormFields();
formFields[0].signatureType = 'Type';
formFields[0].value = 'Syncfusion \n' + new Date();
viewer.updateFormFieldsValue(formFields[0]);
}
</script>
` ``

```

#### **SERVER-BACKED**

```

` ``html
<div>
<div style="width:100%;height:600px">

```

```

<button type="button" onclick="signatureWithDate()">Add Signature with the
time</button>
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
 "~/Home/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Render()
</div>
</div>
<script>
function signatureWithDate() {
var viewer = document.getElementById('pdfviewer').ej2_instances[0];
var formFields = viewer.retrieveFormFields();
formFields[0].signatureType = 'Type';
formFields[0].value = 'Syncfusion \n' + new Date();
viewer.updateFormFieldsValue(formFields[0]);
}
</script>
`

```

### [View Sample in GitHub](#)

#### How to change the default width and height

To change the default width and height of the Syncfusion PDF Viewer, modify the **Width** and **Height** properties of the PdfViewer control.

```

`html
<div>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPa
th("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").Height("1000px").Width("80%").Render()
</div>
</div>
`

```

In this example, the Width property is set to **80%**, and the Height property is set to **1000px**. Change these values to any desired size in pixels or percentages.

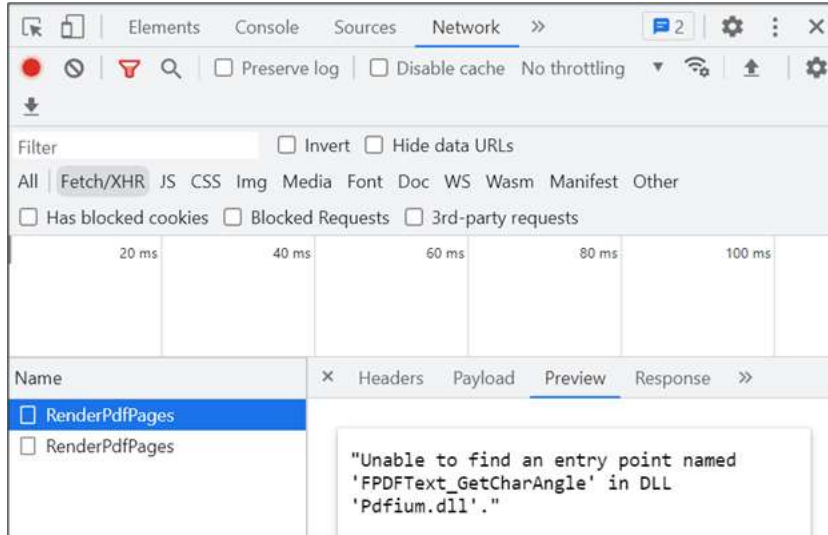
### [View sample in GitHub](#)

#### Resolve "Unable to find an entry point named FPDFText\_GetCharAngle" error

From the release of version **21.1.0.35 (2023 Volume 1)** of Essential Studio, the Pdfium package has been upgraded to improve various functionalities like text search, text selection, rendering, and even performance. If you are updating your project to this version of the Syncfusion PDF Viewer, you may encounter the **"Web-Service is not listening"** error. The Network tab can help you identify the root cause of the issue, which is typically caused by an old version of pdfium assembly being referenced in the local web service project. Below are the assemblies to be referred to in the respective operating systems.

- Windows – pdfium.dll
- Linux – libpdfium.so

- OSX – libpdfium.dylib



*To solve this issue, you should follow the below steps:*

1. Clear the bin and object files of the web-service project.
2. Re-publish the web-service project.

**Note: Note:** If you are hosting your application in Azure, AWS, or in Linux environments, delete the older published files and republish the application.

How to clear the "Web-service is not listening" to error

![Alt text](./images/web-service.png)

If you are facing a **Web-service is not listening** to error in the Syncfusion PDF Viewer, there could be several reasons for this. To troubleshoot the issue, you can use the Network tab in your browser's developer tools to gather more information. Here are the steps you can follow:

**Step 1:** Open the browser's developer tools by right-clicking on the page and selecting **Inspect** from the dropdown menu. Then Navigate to the **Network** tab. This will show you all of the requests that are being made by the page.

![Alt text](./images/networktab.png)

**Step 2:** Try to request the web service. If the service is not listening, the request will fail, and you should see an error message in the Network tab. Click on the failing request to see the details of the error, such as the error message or stack trace. This can help you identify the root cause of the issue. Check the server logs for any errors or warnings that may indicate the cause of the issue and help you to troubleshoot the problem.

**Step 3:** Check the request URL and parameters to see if they are correct. If there is a type or an incorrect parameter, the web service may be unable to process the request.

By following these steps and using the Network tab in your browser's developer tools, you can gather more information about the issue and troubleshoot the problem more effectively.

**Note:** Make sure you are connected to the internet and that your connection is stable. You can try accessing other websites or services to see if they are working, and make sure the URL you are using to access the web service is correct and properly formatted.

*Here are some common exceptions*

- File not found.
- Document cache not found.
- Document pointer does not exist in the cache.

#### *File not found*

If you are encountering an error message stating that the web service is not listening due to a file not being found in the Syncfusion PDF viewer, you can try the following steps to resolve the issue:

##### *Check the file path*

Ensure that the file path you use to access the PDF file is correct and that the file exists in that location. You will need to update the file path if the file does not exist.

##### *Document cache not found*

The **Document cache not found** exception in Syncfusion PDF Viewer typically occurs when the cache used to store the rendered pages of a PDF document is not found or has been deleted. This can happen if the cache directory is changed or deleted or if the application is running in a different environment than it was previously.

##### *Check for multiple instances*

It's possible that you have multiple instances of the Syncfusion PDF Viewer running simultaneously, which can cause issues with the document cache. To check for this, open the Task Manager on your computer and look for any instances of the Syncfusion PDF Viewer running. If you find multiple instances, try closing them all and reopening the viewer.

We can use Redis cache and distributive cache for this issue.

##### *Check your network connection*

Ensure that your network connection is stable and strong enough to support the web service you are trying to use. Sometimes, simply restarting the web service can resolve the issue. Try stopping and starting the service again to see if it resolves the problem.

##### *The document pointer does not exist in the cache.*

The **Document pointer does not exist in the cache** exception in the Syncfusion PDF Viewer usually occurs when there is an issue with loading or caching the PDF document. This error can be caused by a variety of reasons, including:

To clear this error in the Syncfusion PDF Viewer, you can try the following steps:

**Step 1:** Clearing the cache may help resolve the issue. To clear the cache, navigate to the cache location, which can be found in the Syncfusion PDF Viewer's settings or configuration files. Once you locate the cache folder, delete its contents.

**Step 2:** Try reloading the document to ensure it is loaded correctly. You can do this by calling the controller's Load() method. Ensure the document is not already loaded before attempting to load it again.

**Step 3:** Restart the application. If clearing the cache does not work, you can try restarting the PDF Viewer application. This will reload all the necessary components and may resolve the error.

### Internal server error

Server-side exceptions happen for various use cases. We can't just define them if they are document-specific, provide the document, or you may need to contact Syncfusion support for further assistance.

### Load N number of pages on initial loading

The initial rendering in a PDF viewer allows users to control the number of pages displayed when opening a PDF document. This improves the user experience by providing flexibility in loading a specific number of pages initially, while additional pages are dynamically rendered as the user scrolls through the document. This approach enhances the responsiveness of the PDF viewer and minimizes delays as users can access specific pages without waiting for the entire document to load.

To utilize this capability in Syncfusion PDF Viewer, use the [initialRenderPages](#) property. You can achieve the desired outcome by setting this property to the desired number of pages you want to load initially. However, it's important to exercise caution when setting a high value for the initialRenderPages in large documents with numerous pages. Rendering a large number of pages simultaneously can increase memory usage and slow down loading times, impacting the performance of the PDF viewer.

Using the `initialRenderPages` property judiciously is advisable, especially when dealing with larger documents. It is more suitable for scenarios where a smaller range of pages, such as 10-20, can be loaded to provide a quick initial view of the document.

### STANDALONE

```
````cs
@{
    ViewBag.Title = "Home Page";
    double InitialRenderPages = 10;
}
<div>
<div style="height:100%; width: 100%;">
    @Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-
succinctly.pdf").InitialRenderPages(InitialRenderPages).Render()
</div>
</div>
````
```

### SERVER-BACKED

```
````cs
@{
    ViewBag.Title = "Home Page";
    double InitialRenderPages = 10;
}
<div>
<div style="height:100%; width: 100%;">
    @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute(
"~/Home/")).DocumentPath("https://cdn.syncfusion.com/content/pdf/pdf-
succinctly.pdf").InitialRenderPages(InitialRenderPages).Render()
</div>
</div>
````
```

[View sample in GitHub](#)



## Retry Timeout

The [Retry Timeout](#) feature allows you to specify a duration for the PDF Viewer to retry failed AJAX requests before considering them permanent failures. It helps in handling temporary failures due to network issues or server unavailability.

The `retryTimeout` allows developers to specify a duration after which the AJAX request should retry failed requests automatically. Developers can ensure a more resilient and fault-tolerant PDF viewing experience by configuring an appropriate retry timeout value.

By default, when an AJAX request fails, the Retry Timeout property is set to `0`, indicating that no timeout is set. In this case, the PDF Viewer will wait indefinitely for a response, potentially leading to a hanging request. However, you can set the Retry Timeout property to a positive number, specifying the maximum number of seconds the PDF Viewer should wait for a response. If the response is not received within the specified time, the request will be aborted, triggering an appropriate error or timeout property.

To set the retry timeout, use the [retryTimeout](#) property in the PDF Viewer configuration. This property takes a value in seconds.

```
`cs
@{
 ViewBag.Title = "Home Page";

 double RetryTimeout = 10;
 double RetryCount = 5;
}
<div>
<div style="height:100%; width: 100%;">
 @Html.EJS().PdfViewer("pdfviewer").ServiceUrl(VirtualPathUtility.ToAbsolute("~/Home/")).DocumentPath("PDF_Succinctly.pdf").RetryCount(RetryCount).RetryTimeout(RetryTimeout).Render()
</div>
</div>
`
```

In the given example, the [retryTimeout](#) is set to 10 seconds, and the [retryCount](#) is set to 5. This means that if a request made by the PDF Viewer takes longer than 10 seconds to receive a response, it will be considered a timeout. In such cases, The PDF Viewer will resend the same request based on the `retryCount`. Here, this process will repeat up to maximum of 5 retries.

When an exception occurs during the AJAX request in the context of the PDF Viewer, the request will wait for the specified [retryTimeout](#) duration. If the timeout duration is exceeded, the PDF Viewer will decrement the [retryCount](#) and attempt to load the document again. This retry process continues until the document is successfully loaded or the `retryCount` limit is reached.

The [retryCount](#) property of the PDF Viewer allows you to set the number of retries for a specific request. This feature is particularly useful for handling temporary errors such as network timeouts or server

issues. By initiating new requests according to the retry count, ensure a smoother user experience and efficiently handle network or server problems.

If the timeout duration specified by [retryTimeout](#) is exceeded during the AJAX request, the PDF Viewer will decrement the [retryCount](#) and initiate a new request. This process will continue until the document is successfully loaded or the retry count limit is reached. This functionality helps improve the handling of temporary errors and ensures a more efficient and user-friendly experience.

[View sample in GitHub](#)

## Customize toolbar in PDF Viewer component

### *How to customize existing toolbar in PDF Viewer component*

PDF Viewer allows you to customize(add, show, hide, enable, and disable) existing items in a toolbar.

- Add - New items can be defined by **CustomToolbarItemModel** and with existing items in [ToolbarSettings](#) property. Newly added item click action can be defined in [toolbarclick](#).
- Show, Hide - Existing items can be shown or hidden using the [ToolbarSettings](#) property. Pre-defined toolbar items are available with [ToolbarItem](#).
- Enable, Disable - Toolbar items can be enabled or disabled using [enabletoolbaritem](#).

## STANDALONE

```
<div>
<div style="height:500px;width:1350px;">

@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-
succinctly.pdf").ResourceUrl("https://cdn.syncfusion.com/ej2/24.1.41/dist/ej
2-pdfviewer-lib").ToolbarClick("toolbarClick").Render()
</div>
</div>
<script type="text/javascript">
window.onload = function () {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
var toolItem1 = {
prefixIcon: 'e-icons e-paste',
id: 'print',
tooltipText: 'Custom toolbar item',
align: 'left'
};
var toolItem2 = {
id: 'download',
text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
};
var LanguageList = ['Typescript', 'Javascript', 'Angular', 'C#', 'C',
'Python'];
var toolItem3 = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
```

```

template: new ej.dropdowns.ComboBox({ width: 100, value: 'TypeScript',
dataSource: LanguageList, popupWidth: 85, showClearButton: false, readonly:
false })
};
var toolItem4 = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new ej.inputs.TextBox({ width: 125, placeholder: 'Type Here',
created: onCreate })
}; align: 'left'
function onCreate() {
this.addIcon('prepend', 'e-icons e-search');
}
pdfViewer.toolbarSettings = {
showTooltip: true,
toolbarItems: [toolItem1, toolItem2, 'OpenOption', 'PageNavigationTool',
'MagnificationTool', toolItem3, 'PanTool', 'SelectionTool', 'SearchOption',
'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool',
'FormDesignerEditTool', toolItem4, 'CommentTool', 'SubmitForm']
};
};
// Define the toolbarClick event handler
function toolbarClick(args) {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
if (args.item && args.item.id === 'print') {
pdfViewer.printModule.print();
} else if (args.item && args.item.id === 'download') {
pdfViewer.download();
}
}
}
</script>

```

## **SERVER-BACKED**

```

<div>
<div style="height:500px;width:1350px;">

@Html.EJS().PdfViewer("pdfviewer").DocumentPath("https://cdn.syncfusion.com/
content/pdf/pdf-
succinctly.pdf").ServiceUrl(VirtualPathUtility.ToAbsolute("~/api/PdfViewer/"
)).ToolbarClick("toolbarClick").Render()
</div>
</div>
<script type="text/javascript">
window.onload = function () {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
var toolItem1 = {
prefixIcon: 'e-icons e-paste',
id: 'print',
tooltipText: 'Custom toolbar item',
align: 'left'
};
var toolItem2 = {

```

```

id: 'download',
text: 'Save',
tooltipText: 'Custom toolbar item',
align: 'right'
};
var LanguageList = ['Typescript', 'Javascript', 'Angular', 'C#', 'C',
'Python'];
var toolItem3 = {
type: 'Input',
tooltipText: 'Language List',
cssClass: 'percentage',
align: 'Left',
id: 'dropdown',
template: new ej.dropdowns.ComboBox({ width: 100, value: 'TypeScript',
dataSource: LanguageList, popupWidth: 85, showClearButton: false, readonly:
false })
};
var toolItem4 = {
type: 'Input',
tooltipText: 'Text',
align: 'Right',
cssClass: 'find',
id: 'textbox',
template: new ej.inputs.TextBox({ width: 125, placeholder: 'Type Here',
created: onCreate })
}; align: 'left'
function onCreate() {
this.addIcon('prepend', 'e-icons e-search');
}
pdfViewer.toolbarSettings = {
showTooltip: true,
toolbarItems: [toolItem1, toolItem2, 'OpenOption', 'PageNavigationTool',
'MagnificationTool', toolItem3, 'PanTool', 'SelectionTool', 'SearchOption',
'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool',
'FormDesignerEditTool', toolItem4, 'CommentTool', 'SubmitForm']
};
};
// Define the toolbarClick event handler
function toolbarClick(args) {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
if (args.item && args.item.id === 'print') {
pdfViewer.printModule.print();
} else if (args.item && args.item.id === 'download') {
pdfViewer.download();
}
}
}
</script>

```

**Note:** Default value of toolbar items is ['OpenOption', 'PageNavigationTool', 'MagnificationTool', 'PanTool', 'SelectionTool', 'SearchOption', 'PrintOption', 'DownloadOption', 'UndoRedoTool', 'AnnotationEditTool', 'FormDesignerEditTool', 'CommentTool', 'SubmitForm']

#### Align Property

The align property is used to specify the alignment of a toolbar item within the toolbar.

**Left:** Aligns the item to the left side of the toolbar.

**Right:** Aligns the item to the right side of the toolbar.

#### Tooltip Property

The tooltip property is used to set the tooltip text for a toolbar item. Tooltip provides additional information when a user hovers over the item.

#### CssClass Property

The cssClass property is used to apply custom CSS classes to a toolbar item. It allows custom styling of the toolbar item.

#### Prefix Property

The prefix property is used to set the CSS class or icon that should be added as a prefix to the existing content of the toolbar item.

#### ID Property

The id property within a CustomToolbarItemModel is a compulsory attribute that plays a vital role in toolbar customization. It serves as a unique identifier for each toolbar item, facilitating distinct references and interactions.

When defining or customizing toolbar items, it is mandatory to assign a specific and descriptive id to each item.

These properties are commonly used when defining custom toolbar items with the CustomToolbarItemModel in the context of Syncfusion PDF Viewer. When configuring the toolbar using the ToolbarSettings property, you can include these properties to customize the appearance and behavior of each toolbar item.

**Note:** When customizing toolbar items, you have the flexibility to include either icons or text based on your design preference.

[View sample in GitHub](#)

## Troubleshooting

### Document Loading Issues in Version 23.1 or Newer

If you're experiencing problems with your document not rendering in the viewer, especially when using version 23.1 or a newer version, follow these troubleshooting steps to resolve the issue:

1. **Check for viewer.dataBind() Requirement:** Ensure that you have called `viewer.dataBind()` as required in version 23.1 or newer. This explicit call is essential for initializing data binding and document rendering correctly. It is must to call the `dataBind()` method before load.

```
`html
<button id="viewer" onclick="documentLoad()">Load</button>
<div style="width:100%;height:600px">
@Html.EJS().PdfViewer("pdfviewer").Render()
</div>
<script>
function documentLoad () {
var pdfViewer = document.getElementById('pdfviewer').ej2_instances[0];
```

```
pdfViewer.serviceUrl = "https://ej2services.syncfusion.com/angular/development/api/pdfviewer";
pdfViewer.dataBind();
pdfViewer.load("https://cdn.syncfusion.com/content/pdf/annotations.pdf");
}
</script>
,
```

2. **Verify Document Source:** Confirm that the document source or URL you're trying to display is valid and accessible. Incorrect URLs or document paths can lead to loading issues.
3. **Network Connectivity:** Ensure that your application has a stable network connection. Document rendering may fail if the viewer can't fetch the document due to network issues.
4. **Console Errors:** Use your browser's developer tools to check for any error messages or warnings in the console. These messages can provide insights into what's causing the document not to load.
5. **Loading Sequence:** Make sure that you're calling `viewer.dataBind()` and initiating document loading in the correct sequence. The viewer should be properly initialized before attempting to load a document.
6. **Update Viewer:** Ensure that you're using the latest version of the viewer library or framework. Sometimes, issues related to document loading are resolved in newer releases.
7. **Cross-Origin Resource Sharing (CORS):** If you're loading documents from a different domain, ensure that CORS headers are correctly configured to allow cross-origin requests.
8. **Content Security Policies (CSP):** Check if your application's Content Security Policy allows the loading of external resources, as this can affect document loading. Refer [here](#) to troubleshoot.

By following these troubleshooting steps, you should be able to address issues related to document loading in version 23.1 or newer, ensuring that your documents render correctly in the viewer.

## Pivot Table

### Getting Started with ASP.NET MVC Pivot Table Control

This section briefly explains about how to include [ASP.NET MVC Pivot Table](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

##### ~/ \_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

##### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

## Add ASP.NET MVC Pivot Table control

- Now, add the Syncfusion ASP.NET MVC Pivot Table control in `~/Views/Home/Index.cshtml` page.
- The Pivot Table control further needs to be populated with an appropriate data source. For illustration purpose, a collection of objects mentioning the sales details of certain products over a period and region has been prepared. This sample data is assigned to the pivot table control through [DataSource](#) property under [PivotViewDataSourceSettings](#) class.

**CSHTML**

```
@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource((IEnumerable<object>)Model)).Render()
```

**HOMECONTROLLER.CS**

```
public ActionResult Index()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Sold = 31, Amount = 52824, Country = "France",
 Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country = "France",
 Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country = "France",
 Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 27, Amount = 46008, Country = "France",
 Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country = "France",
 Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 127800, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 69, Amount = 117576, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
 "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country = "France",
 Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
 "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 57, Amount = 85448, Country = "France",
 Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country = "France",
 Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
}
```



```

pivotData.Add(new PivotData { Sold = 67, Amount = 70008, Country = "France",
Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 89, Amount = 60496, Country = "France",
Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 75, Amount = 801880, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 57, Amount = 204168, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 75, Amount = 737800, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 87, Amount = 884168, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 39, Amount = 729576, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 90, Amount = 38860, Country = "France",
Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 93, Amount = 139412, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 51, Amount = 92824, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 61, Amount = 76904, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 43360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 85, Amount = 62600, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 97, Amount = 86008, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 69, Amount = 93496, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 49, Amount = 89576, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 40, Amount = 33360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 96, Amount = 77264, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 23, Amount = 24422, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 67, Amount = 75448, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 52345, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 13, Amount = 135612, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 57, Amount = 90008, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 29, Amount = 90496, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });

```

```

pivotData.Add(new PivotData { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 99, Amount = 829576, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 80, Amount = 38360, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 91, Amount = 67824, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 81, Amount = 99904, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 49360, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 65, Amount = 69600, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 57, Amount = 90008, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 29, Amount = 90496, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 85, Amount = 391880, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 97, Amount = 904168, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 85, Amount = 237800, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 77, Amount = 384168, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 99, Amount = 829576, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 80, Amount = 38360, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 76, Amount = 97264, Country = "United
States", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 53, Amount = 94422, Country = "United
States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 90, Amount = 45448, Country = "United
States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 29, Amount = 92345, Country = "United
States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 67, Amount = 235612, Country = "United
States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 97, Amount = 90008, Country = "United
States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 79, Amount = 90496, Country = "United
States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 95, Amount = 501880, Country = "United
States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 97, Amount = 104168, Country = "United
States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 95, Amount = 837800, Country = "United
States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });

```

```

pivotData.Add(new PivotData { Sold = 87, Amount = 684168, Country = "United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 109, Amount = 29576, Country = "United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
return View(pivotData);
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Pivot Table control will be rendered in the default web browser.

#### Adding fields to row, column, value and filter axes

Now that pivot table is initialized and assigned with sample data, will further move to showcase the component by organizing appropriate fields in row, column, value and filter axes.

In [PivotViewDataSourceSettings](#) class, four major axes - [Rows](#), [Columns](#), [Values](#) and [Filters](#) plays a vital role in defining and organizing fields from the bound data source, to render the entire pivot table component in a desired format.

[Rows](#) – Collection of fields that needs to be displayed in row axis of the pivot table.

[Columns](#) – Collection of fields that needs to be displayed in column axis of the pivot table.

[Values](#) – Collection of fields that needs to be displayed as aggregated numeric values in the pivot table.

[Filters](#) - Collection of fields that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each field in the respective axis, the following basic properties should be set.

- [Name](#): It allows to set the field name from the bound data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will not be rendered.
- [Caption](#): It allows to set the field caption, which is the alias name of the field that needs to be displayed in the pivot table.
- [Type](#): It allows to set the summary type of the field. By default, [SummaryTypes.Sum](#) is applied.

In this illustration, "Year" and "Quarter" are added in column, "Country" and "Products" in row, and "Sold" and "Amount" in value section respectively.

#### CSHTML

```

@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
{
 dataSource.DataSource((IEnumerable<object>)Model).ExpandAll(false).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {

```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).Render()

```

|                 | > FY 2015  |             | > FY 2016  |             | >     |
|-----------------|------------|-------------|------------|-------------|-------|
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units |
| > France        | 450        | 714955      | 526        | 1542104     |       |
| > Germany       | 440        | 563515      | 496        | 1772104     |       |
| > United States | 546        | 754515      | 636        | 2263104     |       |
| Grand Total     | 1436       | 2032985     | 1658       | 5577312     |       |

### Applying formatting to a value field

Formatting defines a way in which values should be displayed. For example, format “C” denotes the values should be displayed in currency pattern. To do so, define the [FormatSetting](#) class with its [Name](#) and [Format](#) properties and add it to [PivotViewFormatSettings](#). In this illustration, the [Name](#) property is set as **Amount**, a field from value section and its format is set as currency. Likewise, we can set format for other value fields as well and add it to [PivotViewFormatSettings](#).

**Note:** Only fields from value section, which is in the form of numeric data values are applicable for formatting.

### CSHTML

```

@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)Model).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |

### Enable Grouping Bar

The grouping bar feature automatically populates fields from the bound data source and allows end users to drag fields between different axes such as columns, rows, values, and filters, and alter pivot table at runtime. It also provides option to sort, filter and remove fields. It can be enabled by setting the [ShowGroupingBar](#) property to **true**.

### CSHTML

```
@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>)Model).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true).FormatSettings(
 formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).Render()
```

|                        |                             |             |            |             |
|------------------------|-----------------------------|-------------|------------|-------------|
| Sum of Units Sold ▼ ✕  | Drop filter here            |             |            |             |
| Sum of Sold Amount ▼ ✕ | Year ↑ ▾ ✕    Quarter ↑ ▾ ✕ |             |            |             |
| Country ↑ ▾ ✕          | FY 2015                     |             | FY 2016    |             |
| Products ↑ ▾ ✕         | Units Sold                  | Sold Amount | Units Sold | Sold Amount |
| ▶ France               | 450                         | \$714,955   | 526        | \$1,542,104 |
| ▶ Germany              | 440                         | \$563,515   | 496        | \$1,772,104 |
| ▶ United States        | 546                         | \$754,515   | 636        | \$2,263,104 |
| Grand Total            | 1436                        | \$2,032,985 | 1658       | \$5,577,312 |

### Enable Pivot Field List

The field list allows to add or remove fields and also rearrange the fields between different axes, including column, row, value, and filter along with filter and sort options dynamically at runtime. It can be enabled by setting the [ShowFieldList](#) property to **true**.

### CSHTML

```
@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Render()
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |

### Calculated field

The calculated field feature allows user to insert or add a new calculated field based on the available fields from the bound data source using basic arithmetic operators. The calculated field can be included in pivot table using the [CalculatedFieldSetting](#) class from code behind. Or else, calculated fields can be added at run time through the built-in dialog by just setting the [AllowCalculatedField](#) property to **true** in pivot table. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation.

**Note:** Calculated field is applicable only for value fields.

### CSHTML

```
@model List<PivotTableSample.Controllers.PivotData>
@{var amount = "\" + "Sum(Amount)" + "\"; }
@{var sold = "\" + "Sum(Sold)" + "\"; }
@{ var totalPrice = amount + "+" + sold; }
@Html.EJS().Button("calculated-field-btn").Content("Calculated
Field").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource
=>
dataSource.DataSource((IEnumerable<object>)Model).ExpandAll(false).EnableSort
ing(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(
10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).CalculatedFieldSettings(calculatedfieldsettings =>
{
 calculatedfieldsettings.Name("Total").Formula(totalPrice).Add();
}))).AllowCalculatedField(true).Render()
<script>
 document.getElementById("calculated-field-
btn").addEventListener('click', function () {
```

```

 var pivotObj =
document.getElementById("PivotView").ej2_instances[0];
 pivotObj.createCalculatedFieldDialog();
 });
</script>

```

| Calculated Field |            |             |            |             |      |
|------------------|------------|-------------|------------|-------------|------|
|                  | > FY 2015  |             | > FY 2016  |             | >    |
|                  | Units Sold | Sold Amount | Units Sold | Sold Amount | Unit |
| > France         | 450        | \$714,955   | 526        | \$1,542,104 |      |
| > Germany        | 440        | \$563,515   | 496        | \$1,772,104 |      |
| > United States  | 546        | \$754,515   | 636        | \$2,263,104 |      |
| Grand Total      | 1436       | \$2,032,985 | 1658       | \$5,577,312 |      |

**Note:** [View Sample in GitHub.](#)

## Data Binding in ASP.NET MVC Pivot Table Component

### JSON

For JSON data binding, the `type` property under [PivotViewDataSourceSettings](#) needs to be set as `JSON`. By default, the default value is assumed as `JSON`.

### Binding JSON data via local

In-order to bind local JSON data to the pivot table user can assign the local variable holding the JSON data to the [DataSource](#) property under [PivotViewDataSourceSettings](#).

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()

```

### LOCALDATA.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
}

```



```
return View();
}
```

Using local variable, the JSON data can also be bound to the pivot table using [DataManager](#) option with the help of [JsonAdaptor](#). Here the instance of [DataManager](#) holding JSON data is assigned to [DataSource](#) property under [PivotViewDataSourceSettings](#). The use of [DataManager](#) is optional here.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Json(ViewBag.dataSource.ToArray()).Adaptor("JsonAdaptor");
}).ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### LOCALJSONDATAMANAGER.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |             | Grand Total |
|----------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|-------------|
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  |
| France         | 729        | \$1,160,100 | 609        | \$983,317   | 703        | \$1,140,998 | 68         | \$108,402   |             |
| Germany        | 528        | \$845,472   | 667        | \$1,067,220 | 579        | \$945,569   | 130        | \$211,903   |             |
| United Kingdom | 782        | \$1,263,110 | 640        | \$1,031,631 | 657        | \$1,041,051 | 161        | \$265,367   |             |
| United States  | 682        | \$1,085,399 | 480        | \$770,362   | 644        | \$1,022,552 | 232        | \$366,358   |             |
| Grand Total    | 2721       | \$4,354,080 | 2396       | \$3,852,530 | 2583       | \$4,150,169 | 591        | \$952,029   |             |

In the meantime, the JSON data from the local \*.json file type can also be connected to the pivot table via the file uploader option. Here, the resulting string after uploading the file needs to be converted to JSON data that can be assigned to the [DataSource](#) property under [PivotViewDataSourceSettings](#). The following code example illustrates the same.

```
`html
```

```
@using Syncfusion.EJ2.PivotView
```

```

@Html.EJS().Uploader("fileupload").Render()
@Html.EJS().PivotView("pivotview").Render()
<script>
// Step 1: Initiate the file uploader
var uploadObj = new.Uploader({});
uploadObj.appendTo('#fileupload');
var input = document.querySelector('input[type="file"]');
// Step 2: Add the event listener which fires when the *.JSON file is uploaded.
input.addEventListener('change', function (e) {
// Step 3: Initiate the file reader
var reader = new FileReader();
reader.onload = function () {
// Step 4: Getting the string output which is to be parsed as JSON.
var result = JSON.parse(reader.result);
var pivotObj = document.getElementById('pivotview');
pivotObj.dataSourceSettings = {
// Step 5: The JSON result to be bound as data source.
dataSource: result
// Step 6: The appropriate report needs to be provided here.
}
}
reader.readAsText(input.files[0]);
});
</script>

```

#### *Binding JSON data via remote*

In-order to bind remote JSON data, mention the endpoint [URL](#) under [PivotViewDataSourceSettings](#) property. The [URL](#) property supports both direct downloadable file (\*.json) and web service URL.

#### **CSHTML**

```

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.Url("https://cdn.syncfusion.com/data/sales-analysis.json").ExpandAll(false).Rows(rows =>
{
 columns.Name("EnerType").Add();
}).Columns(columns =>
{

```

```

 columns.Name("EneSource").Add();
 }).Values(values =>
 {
 values.Name("PowUnits").Add();
 values.Name("ProCost").Add();
 })
).Render()

```

## REMOTEJSONDATA.CS

```

public ActionResult Index()
{
 return View();
}

```

|             | Bio-diesel  |           | Ethanol Fuel |           | Geo-thermal |           | Hydro-electric |           | Solar     |
|-------------|-------------|-----------|--------------|-----------|-------------|-----------|----------------|-----------|-----------|
|             | Units (GWh) | Cost (MM) | Units (GWh)  | Cost (MM) | Units (GWh) | Cost (MM) | Units (GWh)    | Cost (MM) | Units (G) |
| Biomass     | 1042        | 1439      | 595          | 1031      |             |           |                |           |           |
| Free Energy |             |           |              |           | 1528        | 2115      | 3378           | 3244      |           |
| Grand Total | 1042        | 1439      | 595          | 1031      | 1528        | 2115      | 3378           | 3244      |           |

## CSV

For CSV data binding, the `type` property under [PivotViewDataSourceSettings](#) needs to be set as `CSV` mandatorily.

**Note:** The CSV format is considered to be the most compact format compared to JSON since it is half the size of JSON. This helps to reduce the bandwidth while transferring to the browser.

### Binding CSV data via local

In-order to bind local CSV data to the pivot table, user needs to convert it as string array and then directly assign it to the [DataSource](#) property under [PivotViewDataSourceSettings](#).

## CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource
=> dataSource.DataSource('getCSVData()')
.Type(Syncfusion.EJ2.PivotView.DataSourceType.CSV)
.Rows(rows =>
{
 rows.Name("Region").Add(); rows.Name("Country").Add();
}).Columns(columns =>
{
 columns.Name("Item Type").Add();
 columns.Name("Sales Channel").Add();
}).Values(values =>
{
 values.Name("Total Cost").Add(); values.Name("Total Revenue").Add();
 values.Name("Total Profit").Add();
})).Render()
<script>
function getCSVData() {
 var dataSource = [];
 var jsonObject = window.csvdata.split(/\r?\n|\r/);
 for (var i = 0; i < jsonObject.length; i++) {

```

```

 if (!ej.base.IsNullOrEmpty(jsonObject[i]) && jsonObject[i] !=
 '') {
 dataSource.push(jsonObject[i].split(','));
 }
 }
 return dataSource;
}

window.csvdata = "Region,Country,Item Type,Sales Channel,Order
Priority,Order Date,Order ID,Ship Date,Units Sold,Unit Price,Unit Cost,Total
Revenue,Total Cost,Total Profit\r\nMiddle East and North
Africa,Libya,Cosmetics,Offline,M,10/18/2014,686800706,10/31/2014,8446,437.20
,263.33,3692591.20,2224085.18,1468506.02\r\nNorth
America,Canada,Vegetables,Online,M,11/7/2011,185941302,12/8/2011,3018,154.06
,90.93,464953.08,274426.74,190526.34\r\nMiddle East and North
Africa,Libya,Baby
Food,Offline,C,10/31/2016,246222341,12/9/2016,1517,255.28,159.42,387259.76,2
41840.14,145419.62\r\nAsia,Japan,Cereal,Offline,C,4/10/2010,161442649,5/12/2
010,3322,205.70,117.11,683335.40,389039.42,294295.98\r\nSub-Saharan
Africa,Chad,Fruits,Offline,H,8/16/2011,645713555,8/31/2011,9845,9.33,6.92,91
853.85,68127.40,23726.45\r\nEurope,Armenia,Cereal,Online,H,11/24/2014,683458
888,12/28/2014,9528,205.70,117.11,1959909.60,1115824.08,844085.52\r\nSub-
Saharan
Africa,Eritrea,Cereal,Online,H,3/4/2015,679414975,4/17/2015,2844,205.70,117.
11,585010.80,333060.84,251949.96\r\nEurope,Montenegro,Clothes,Offline,M,5/17
/2012,208630645,6/28/2012,7299,109.28,35.84,797634.72,261596.16,536038.56\r\
nCentral America and the
Caribbean,Jamaica,Vegetables,Online,H,1/29/2015,266467225,3/7/2015,2428,154.
06,90.93,374057.68,220778.04,153279.64\r\nAustralia and
Oceania,Fiji,Vegetables,Offline,H,12/24/2013,118598544,1/19/2014,4800,154.06
,90.93,739488.00,436464.00,303024.00,4925394.54,1624319.73\r\nEurope,Portuga
l,Cereal,Offline,C,4/10/2014,811546599,5/8/2014,3528,205.70,117.11,725709.60
,413164.08,312545.52\r\n";
</script>

```

### LOCALCSVDATA.CS

```

public ActionResult Index()
{
 var data = getCSVData();
 ViewBag.DataSource = data;
 return View();
}

```

|                           | Baby Food    |                |              | Beverages   |                |              | Cereal       |                |       |
|---------------------------|--------------|----------------|--------------|-------------|----------------|--------------|--------------|----------------|-------|
|                           | Total Cost   | Total Reven... | Total Profit | Total Cost  | Total Reven... | Total Profit | Total Cost   | Total Reven... | Total |
| Asia                      | \$8,897,230  | \$14,247,177   | \$5,349,947  | \$2,683,076 | \$4,004,780    | \$1,321,704  | \$3,825,749  | \$6,779,808    |       |
| Australia and Oceania     | \$3,435,182  | \$5,500,773    | \$2,065,591  | \$1,096,310 | \$1,636,361    | \$540,051    | \$5,058,566  | \$8,885,212    |       |
| Central America and th... | \$5,596,120  | \$8,961,094    | \$3,364,974  | \$1,806,085 | \$2,695,777    | \$889,692    | \$3,915,222  | \$6,876,962    |       |
| Europe                    | \$20,565,658 | \$32,931,886   | \$12,366,228 | \$3,958,141 | \$5,907,952    | \$1,949,811  | \$11,151,214 | \$19,586,754   |       |
| Middle East and North ... | \$8,192,594  | \$13,118,839   | \$4,926,245  | \$2,267,962 | \$3,385,178    | \$1,117,216  | \$5,696,582  | \$10,005,865   |       |
| North America             | \$2,586,908  | \$4,142,429    | \$1,555,520  | \$220,845   | \$329,635      | \$108,790    |              |                |       |
| Sub-Saharan Africa        | \$20,331,949 | \$32,557,645   | \$12,225,697 | \$4,018,510 | \$5,998,060    | \$1,979,549  | \$15,761,952 | \$27,685,369   | \$... |

In the meantime, the CSV data from the local \*.csv file type can also be connected to the pivot table via the file uploader option. Here, the resulting string after uploading the file needs to be converted to string array that can be assigned to the [DataSource](#) property under [PivotViewDataSourceSettings](#). The following code example illustrates the same.

```
`html
@using Syncfusion.EJ2.PivotView
@Html.EJS().Uploader("fileupload").Render()
@Html.EJS().PivotView("pivotview").Render()
<script>
// Step 1: Initiate the file uploader
var uploadObj = new Uploader({});
uploadObj.appendTo('#fileupload');
var input = document.querySelector('input[type="file"]');
// Step 2: Add the event listener which fires when the *.CSV file is uploaded.
input.addEventListener('change', function (e) {
// Step 3: Initiate the file reader
var reader = new FileReader();
reader.onload = function () {
// Step 4: Getting the string output which is to be converted as string[][]
var result = reader.result.split('\n').map(function (line) {
return line.split(',');
});
var pivotObj = document.getElementById('pivotview');
pivotObj.dataSourceSettings = {
// Step 5: The string[][] result to be bound as data source
dataSource: result,
type: 'CSV'
// Step 6: The appropriate report needs to be provided here.
}
};
reader.readAsText(input.files[0]);
});
</script>
`
```

### Binding CSV data via remote

In-order to bind remote CSV data, mention the endpoint [URL](#) under [PivotViewDataSourceSettings](#) property. The [URL](#) property supports both direct downloadable file (\*.csv) and web service URL.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.Url("https://bi.syncfusion.com/productservice/api/sales")
.ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true).Type(Syncfusion.EJ2.PivotView.DataSourceType.CSV)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Total Cost").Format("C0").UseGrouping(true).Add();
 formatsettings.Name("Total Revenue").Format("C0").UseGrouping(true).Add();
 formatsettings.Name("Total Profit").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Region").Add(); rows.Name("Country").Add();
}).Columns(columns =>
{
 columns.Name("Item Type").Add();
 columns.Name("Sales Channel").Add();
}).Values(values =>
{
 values.Name("Total Cost").Add(); values.Name("Total Revenue").Add();
 values.Name("Total Profit").Add();
})
).Render()
```

### REMOTECSVDATA.CS

```
public ActionResult Index()
{
 return View();
}
```

|                                   | Baby Food    |               |              | Beverages   |               |              | Cereal       |               |              |
|-----------------------------------|--------------|---------------|--------------|-------------|---------------|--------------|--------------|---------------|--------------|
|                                   | Total Cost   | Total Revenue | Total Profit | Total Cost  | Total Revenue | Total Profit | Total Cost   | Total Revenue | Total Profit |
| Asia                              | \$8,997,230  | \$14,247,177  | \$5,349,947  | \$2,683,076 | \$4,004,780   | \$1,321,704  | \$3,825,749  | \$6,719,808   | \$2,894,059  |
| Australia and Oceania             | \$3,435,182  | \$5,500,773   | \$2,065,591  | \$1,096,310 | \$1,636,361   | \$540,051    | \$5,058,566  | \$8,885,212   | \$3,826,646  |
| Central America and the Caribbean | \$5,596,120  | \$8,961,094   | \$3,364,974  | \$1,806,085 | \$2,695,777   | \$889,692    | \$3,915,222  | \$6,876,962   | \$2,961,740  |
| Europe                            | \$20,565,658 | \$32,931,886  | \$12,366,228 | \$3,958,141 | \$5,907,952   | \$1,949,811  | \$11,151,214 | \$19,586,754  | \$8,435,540  |
| Middle East and North Africa      | \$8,192,594  | \$13,118,839  | \$4,926,245  | \$2,267,962 | \$3,385,178   | \$1,117,216  | \$5,696,582  | \$10,005,865  | \$4,309,283  |
| North America                     | \$2,586,900  | \$4,142,429   | \$1,555,529  | \$220,845   | \$329,635     | \$108,790    |              |               |              |
| Sub-Saharan Africa                | \$20,331,949 | \$32,557,645  | \$12,225,697 | \$4,018,510 | \$5,998,060   | \$1,979,549  | \$15,761,952 | \$27,685,369  | \$11,923,417 |

### Remote Data Binding

To interact with remote data source, provide the endpoint [Url](#) within [DataManager](#). By default, [DataManager](#) uses [ODataAdaptor](#) for remote data-binding.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
 }).Columns(columns =>
 {
 columns.Name("ProductName").Caption("Product Name").Add();
 }).Values(values =>
 {
 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
 })
}).Render()
```

### **REMOTEDATA.CS**

```
public ActionResult Index()
{
 return View();
}
```

### *Binding with OData services*

OData is a standardized protocol for creating and consuming data. User can retrieve data from OData service using the **DataManager** class. Refer to the following code example for remote data binding using OData service.

### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders/").CrossDomain(true).Adaptor("ODataAdaptor");
}).ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("CustomerID").Caption("Customer ID").Add();
}).Values(values =>
{
 values.Name("Freight").Caption("Freight").Add();
}))
```

```
.ShowFieldList(true).Render()
```

### ODATA.CS

```
public ActionResult Index()
{
 return View();
}
```

#### *Binding with OData V4 services*

The OData V4 is an improved version of OData protocols, and the **DataManager** class can be used to retrieve and consume OData V4 services. For more details on OData V4 services, refer to the [OData documentation](#). To bind OData V4 service, use the [ODataV4Adaptor](#).

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/Orders/").CrossDomain(true).Adaptor("ODataV4Adaptor");
}).ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
 }).Columns(columns =>
 {
 columns.Name("CustomerID").Caption("Customer ID").Add();
 }).Values(values =>
 {
 values.Name("Freight").Caption("Freight").Add();
 //values.Name("UnitPrice").Caption("Unit Price").Add();
 })
).ShowFieldList(true).Render()
```

### ODATAV4.CS

```
public ActionResult Index()
{
 return View();
}
```

#### *Web API*

User can use **WebApiAdaptor** to bind pivot table with Web API created using OData endpoint.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
```



```

 }).ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
 }).Columns(columns =>
 {
 columns.Name("ProductName").Caption("Product Name").Add();
 }).Values(values =>
 {
 values.Name("Quantity").Caption("Quantity").Add();
values.Name("UnitPrice").Caption("Unit Price").Add();
 })
).Render()

```

### WEB-API.CS

```

public ActionResult Index()
{
 return View();
}

```

### Querying in Data Manager

By default, the data manager retrieves all the data from the provider which is mapped in it. The data from the provider can be filtered, sorted, paged, etc. by setting the own query in `defaultQuery` property in the data manager instance.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
dataManger.Url("https://js.syncfusion.com/demos/ejServices/Wcf/Northwind.svc/Orders").CrossDomain(true).Adaptor("ODataAdaptor");
}).ExpandAll(false).ShowAggregationOnValueField(false).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
 }).Columns(columns =>
 {
 columns.Name("CustomerID").Caption("Customer ID").Add();
 }).Values(values =>
 {
 values.Name("Freight").Caption("Freight").Add();
 })
).Load("load").Render()
<script>
function load(args) {
 var dataSource = args.dataSourceSettings.dataSource;
 dataSource.defaultQuery = new ej.data.Query().take(2);
}

```

```
</script>
```

### ODATAADAPTOR.CS

```
public ActionResult Index()
{
 return View();
}
```

### Mapping

One can define field information like alias name (caption), data type, aggregation type, show and hide subtotals etc. using the [FieldMapping](#) property under [PivotViewDataSourceSettings](#). The available options are,

- [Name](#) - It is to specify the appropriate field name.
- [Caption](#) - It is to set the alias name (caption) to the specific field. Instead of actual field name, the alias name (caption) will be set in the UI of the pivot table.
- [Type](#) - It is to display values in the pivot table with appropriate aggregation such as sum, product, count, average, minimum, maximum, etc. Its default value is **sum**. This option is applicable only for relational data source.
- [Axis](#) - It will help to display the field in specified axis such as row/column/value/filter axis of the pivot table.
- [ShowNoDataItems](#) - It is to show all the members of a specific field to the pivot table, even if there are no data in the intersection of the row and column. The default value is **false**. This option is applicable only for relational data source.
- [BaseField](#) - For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective field is assigned for comparison via this property.
- [BaseItem](#) - For the aggregate types like "DifferenceFrom" or "PercentageOfDifferenceFrom" or "PercentageOfParentTotal", selective member in a field is assigned for comparison via this property.
- [ExpandAll](#) - It is to expand or collapse all headers of a specific field in row and column axes of the pivot table. The default value is **false**.
- [ShowSubTotals](#) - It is to show or hide sub-totals of a specific field in row and column axis of the pivot table. The default value is **true**.
- [IsNamedSet](#) - It is to set whether the specified field is named set or not. In general, the named set is a set of dimension members or a set expression (MDX query) to be created as a dimension in the SSAS OLAP cube itself. The default value is **false** and this option is applicable only for OLAP data source.
- [IsCalculatedField](#) - It is to set whether the specified field is a calculated field or not. In general, a calculated field is created from the bound data source or using simple formula with basic arithmetic operators in the pivot table. The default value is **false** and this option is applicable only for OLAP data source.
- [ShowFilterIcon](#) - It is to show or hide the filter icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This filter icon is used to filter the members of a specified field at runtime in the pivot table. The default value is **true**.
- [ShowSortIcon](#) - It is to show or hide the sort icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This sort icon is used to order members of a specified field either in ascending or descending at runtime. The default value is **true**.

- [ShowRemoveIcon](#) - It is to show or hide the remove icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This remove icon is used to remove the specified field during runtime. The default value is **true**.
- [ShowValueTypeIcon](#) - It is to show or hide the value type icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This value type icon helps to select the appropriate aggregation type to specified value field at runtime. The default value is **true**.
- [ShowEditIcon](#) - It is to show or hide the edit icon of a specific field which will be displayed on the button of the grouping bar and field list UI. This edit icon is used to modify caption, formula, and format of a specified calculated field at runtime. The default value is **true**.
- [AllowDragAndDrop](#) - It is to restrict specific field's button from being dragged on runtime in the grouping bar and field list UI. This will prevent from altering the current report. The default value is **true**.
- [DataType](#) - It is to specify the type of the field like 'string', 'number', 'datetime', 'date', and 'boolean'.
- [GroupName](#) - It is to display fields in the field list UI by grouping them under the desired folder name.

The main purpose of these mapping options is to configure each field that is not part of the initial pivot report. Even if any field that is part of this mapping is defined here, the value set in the initial report will have the highest preceding.

**Note:** This option is applicable only for relational data source.

In the below code sample, visibility of the field button icons are configured.

#### CSHTML

```
@using Syncfusion.EJ2
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
dataSource =>
{
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true).AllowValueFilter(true).FormatSettings(
formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Production Year").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
})).FieldMapping(fields=>
{
 fields.Name("Quarter").ShowSortIcon(false).Add();
 fields.Name("Products").ShowFilterIcon(false).ShowRemoveIcon(false).Add();
 fields.Name("Amount").ShowValueTypeIcon(false).Caption("Sold Amount").Add();
}).ShowGroupingBar(true).ShowFieldList(true).Render()
```

**FIELD\_MAPPING.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Sold = 31, Amount = 52824, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 27, Amount = 46008, Country = "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country = "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country = "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country = "France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 127800, Country = "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country = "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 69, Amount = 117576, Country = "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country = "France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country = "France", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country = "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 57, Amount = 85448, Country = "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country = "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 93, Amount = 139412, Country = "France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 35, Amount = 52470, Country = "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 28, Amount = 41977, Country = "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 48, Amount = 71957, Country = "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 36, Amount = 53969, Country = "France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 37480, Country = "France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
}

```

```

pivotData.Add(new PivotData { Sold = 69, Amount = 103436, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 16, Amount = 23989, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 28, Amount = 41977, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 19, Amount = 28486, Country =
"France", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 89, Amount = 141999.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 91, Amount = 145190.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 24, Amount = 38292, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 75, Amount = 119662.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 100, Amount = 159550, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 30, Amount = 47865, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 69, Amount = 110089.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 25, Amount = 39887.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 42, Amount = 67011, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 94, Amount = 149977, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 76, Amount = 121258, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 52, Amount = 82966, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 33, Amount = 52651.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 16, Amount = 23989, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 21, Amount = 33505.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 74, Amount = 126096, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 99, Amount = 148406, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 31, Amount = 49460.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 57, Amount = 97128, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 41, Amount = 61464, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 64, Amount = 102112, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 85, Amount = 144840, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 76, Amount = 129504, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 33, Amount = 56232, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });

```

```

pivotData.Add(new PivotData { Sold = 71, Amount = 120984, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 81, Amount = 138024, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 65, Amount = 110760, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 39, Amount = 66456, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 91, Amount = 155064, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 59, Amount = 100536, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 36, Amount = 61344, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 39, Amount = 58466, Country =
"Germany", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 47, Amount = 70458, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 19, Amount = 28486, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 34, Amount = 50971, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 34, Amount = 50971, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 26, Amount = 38979, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 15, Amount = 22490, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 79, Amount = 118426, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 14, Amount = 20991, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 23932.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 47, Amount = 74988.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 93, Amount = 148381.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 13, Amount = 20741.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 44, Amount = 70202, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 59, Amount = 94134.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 34, Amount = 54247, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 48, Amount = 76584, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 35, Amount = 55842.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 71, Amount = 113280.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q4" });

```

```

 pivotData.Add(new PivotData { Sold = 77, Amount = 131208, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 92, Amount = 156768, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 91, Amount = 155064, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 56, Amount = 95424, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 14, Amount = 23856, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 24, Amount = 40896, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 39, Amount = 66456, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 84, Amount = 143136, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 40, Amount = 68160, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 96, Amount = 163584, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2018", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 24, Amount = 35981, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 86, Amount = 128919, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 31, Amount = 46474, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 36, Amount = 53969, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 40, Amount = 59965, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 69, Amount = 103436, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"
});

```



```
 pivotData.Add(new PivotData { Sold = 95, Amount = 142410, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 95, Amount = 142410, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 30, Amount = 44975, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 11, Amount = 16494, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 97, Amount = 145408, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 16, Amount = 23989, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 40, Amount = 59965, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 68, Amount = 101937, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 11, Amount = 16494, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 27, Amount = 40478, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 45, Amount = 67460, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 100, Amount = 149905, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 70, Amount = 104935, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 100, Amount = 149905, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 18, Amount = 26987, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 70, Amount = 104935, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 81, Amount = 121424, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1"
});
```



```

 pivotData.Add(new PivotData { Sold = 99, Amount = 148406, Country =
"United States", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 43, Amount = 73272, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 43, Amount = 73272, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 52, Amount = 88608, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 91, Amount = 155064, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 37, Amount = 63048, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 41, Amount = 69864, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 23, Amount = 39192, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 85, Amount = 144840, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 28, Amount = 47712, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 53, Amount = 90312, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2018", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 82, Amount = 130831, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 41, Amount = 65415.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 60, Amount = 95730, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 71, Amount = 113280.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 45, Amount = 71797.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q3" });

```

```
 pivotData.Add(new PivotData { Sold = 21, Amount = 33505.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 94, Amount = 149977, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 34, Amount = 54247, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 14, Amount = 22337, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 76, Amount = 121258, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 50, Amount = 79775, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 119662.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 78179.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 40, Amount = 63820, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 94, Amount = 149977, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 17, Amount = 27123.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 45, Amount = 71797.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 56, Amount = 89348, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 119662.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 11, Amount = 17550.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 54, Amount = 86157, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 14, Amount = 22337, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 11, Amount = 17550.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 76, Amount = 121258, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q4" });
```

```

 pivotData.Add(new PivotData { Sold = 45, Amount = 71797.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2018", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 80, Amount = 127640, Country =
"United States", Products = "Touring Bikes", Year = "FY 2018", Quarter =
"Q1" });
 return pivotData;
 }
}
public class PivotData
{
 public int Sold { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string Products { get; set; }
 public string Year { get; set; }
 public string Quarter { get; set; }
}

```

|                   |                  |         |         |         |             |
|-------------------|------------------|---------|---------|---------|-------------|
| Sum of Units Sold | Drop filter here |         |         |         |             |
|                   | Year             |         |         |         |             |
| Country           | FY 2015          | FY 2016 | FY 2017 | FY 2018 | Grand Total |
| France            | 46539            | 44513   | 48207   | 11516   | 151175      |
| Germany           | 44154            | 47910   | 47214   | 11419   | 150697      |
| United Kingdom    | 25457            | 22944   | 25836   | 6375    | 81612       |
| United States     | 47724            | 47691   | 46602   | 11715   | 153732      |
| Grand Total       | 164274           | 164058  | 167859  | 41025   | 537216      |

### Values in row axis

By default, the value fields are plotted in column axis. To plot those fields in row axis, use the `valueAxis` property by setting its value as `row`. By default, it holds the value `column`.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ValueAxis("row")
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()

```

### VALUESINROW.CS

```

public ActionResult Index()
{

```

```

var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```

|                 | ► FY 2015 | ► FY 20...  | ► FY 20...  | ► FY 20... | Grant |
|-----------------|-----------|-------------|-------------|------------|-------|
| ► France        |           |             |             |            |       |
| Units Sold      | 450       | 526         | 592         | 16         |       |
| Sold Amount     | \$714,955 | \$1,542,104 | \$2,903,308 | \$27,264   | \$5,  |
| ► Germany       |           |             |             |            |       |
| Units Sold      | 440       | 496         | 372         | 96         |       |
| Sold Amount     | \$563,515 | \$1,772,104 | \$1,634,808 | \$77,264   | \$4,  |
| ► United States |           |             |             |            |       |

#### Values at different positions

By default, the value fields are placed at the end of the row or column axis. To place those value fields in different positions, use the [valueIndex](#) property and set the value to an appropriate index position. Its default value is **-1**, which denotes the last position. The [valueIndex](#) property is dependent on the [valueAxis](#) property.

**Note:** This support is only available for relational data sources. Also, enable the [showValuesButton](#) property in the grouping bar and field list UI to **true** to re-arrange the values fields at different positions via user interaction.

#### CSHTML

```

@Html.EJS().PivotView("PivotView").Width("100%").Height("350").DataSourceSet
tings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
).ValueIndex(1)
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Add();
 })
).Render()

```

**MEASUREATDIFFERENTPOSITION.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                | ▼ FY 2015  |     |     |     |
|----------------|------------|-----|-----|-----|
|                | Units Sold |     |     |     |
|                | Q1         | Q2  | Q3  | Q4  |
| ▼ France       | 114        | 108 | 110 | 118 |
| Mountain Bikes | 31         | 51  | 90  | 25  |
| Road Bikes     | 83         | 57  | 20  | 93  |
| ▼ Germany      | 74         | 128 | 140 | 98  |
| Mountain Bikes | 51         | 61  | 70  | 85  |
| Road Bikes     | 23         | 67  | 70  | 13  |

**Show 'no data' items**

By default, the pivot table only shows the field item if it has data in its row or column combination. To show all items that do not have data in row and column combination in the pivot table, use the [ShowNoDataItems](#) property by settings its value to **true** for the desired fields. In the following code sample, rows of the "Country" and "Products" fields do not have data in all combination with "Year" and "Quarter" column field.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").ShowNoDataItems(true).Add();
 rows.Name("Products").ShowNoDataItems(true).Add();
 }).Columns(columns =>
 {
 columns.Name("Year").ShowNoDataItems(true).Add();
 columns.Name("Quarter").ShowNoDataItems(true).Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()
```

**NODATA.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Amount = 100, Country = "Canada", Date = "FY 2005", Products = "Bike", Quantity = 2, State = "Alberta" });
 pivotData.Add(new PivotData { Amount = 200, Country = "Canada", Date = "FY 2006", Products = "Van", Quantity = 3, State = "British Columbia" });
 pivotData.Add(new PivotData { Amount = 150, Country = "United States", Date = "FY 2006", Products = "Car", Quantity = 3, State = "New Mexico" });
 pivotData.Add(new PivotData { Amount = 200, Country = "United States", Date = "FY 2005", Products = "Bike", Quantity = 4, State = "New York" });
 return pivotData;
}

public class PivotData
{
 public int Quantity { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string Products { get; set; }
 public string Date { get; set; }
 public string State { get; set; }
}

```

|                | FY 2018    |             |            |             |       |
|----------------|------------|-------------|------------|-------------|-------|
|                | Q1         |             | Q2         |             | Q3    |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units |
| France         | 16         | \$27,264    |            |             |       |
| Mountain Bikes | 16         | \$27,264    |            |             |       |
| Road Bikes     |            |             |            |             |       |
| Germany        | 96         | \$77,264    |            |             |       |
| Mountain Bikes | 96         | \$77,264    |            |             |       |

Always shows the value headers

To show the value header always in pivot table even it holds a single value, use the [AlwaysShowValueHeader](#) property by settings its value as **true**.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).AlwaysShowValueHeader(true)
 .Rows(rows =>

```

```

{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}) .Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}) .Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
})) .Render()

```

### SINGLE-CALCULATION-HEADER.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                 | ► FY 2015   | ► FY 2016   | ► FY 2017   | ► FY 2018   | Grand Total |
|-----------------|-------------|-------------|-------------|-------------|-------------|
|                 | Sold Amount | Sold Amount | Sold Amount | Sold Amount | Sold Amount |
| ► France        | \$714,955   | \$1,542,104 | \$2,903,308 | \$27,264    | \$5,187     |
| ► Germany       | \$563,515   | \$1,772,104 | \$1,634,808 | \$77,264    | \$4,047     |
| ► United States | \$754,515   | \$2,263,104 | \$3,387,308 | \$97,264    | \$6,502     |
| Grand Total     | \$2,032,985 | \$5,577,312 | \$7,925,424 | \$201,792   | \$15,737    |

### Customize empty value cells

User can show custom string in empty value cells using the [EmptyCellsTextContent](#) property in [PivotViewDataSourceSettings](#) class of the pivot table. Since the property is of string data type, user can fill empty value cells with any value like "0", "-", "\*", "(blank)", etc. Its common for all value fields and can be configured through code behind.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).EmptyCellsTextContent("**")
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}) .Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}) .Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}

```

```
}).Render()
```

### EMPTY-CELLS.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Amount = 100, Country = "Canada", Date =
 "FY 2005", Products = "Bike", Quantity = 2, State = "Alberta" });
 pivotData.Add(new PivotData { Amount = 200, Country = "Canada", Date =
 "FY 2006", Products = "Van", Quantity = 3, State = "British Columbia" });
 pivotData.Add(new PivotData { Amount = 150, Country = "United States",
 Date = "FY 2006", Products = "Car", Quantity = 3, State = "New Mexico" });
 pivotData.Add(new PivotData { Amount = 200, Country = "United States",
 Date = "FY 2005", Products = "Bike", Quantity = 4, State = "New York" });
 return pivotData;
}
public class PivotData
{
 public int Quantity { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string Products { get; set; }
 public string Date { get; set; }
 public string State { get; set; }
}
```

|                | FY 2015    |             |            |             |            |
|----------------|------------|-------------|------------|-------------|------------|
|                | Q1         |             | Q2         |             | Q3         |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France         | 114        | \$177,246   | **         | **          | 110        |
| Mountain Bikes | 31         | \$52,824    | **         | **          | 90         |
| Road Bikes     | 83         | \$124,422   | **         | **          | 20         |
| Germany        | 74         | \$117,246   | 128        | \$152,352   | 140        |
| Mountain Bikes | 51         | \$92,824    | 61         | \$76,904    | 70         |

### Events

#### Load

The event **Load** fires before initiate rendering of pivot table. In this event user can customize data source settings before initiating pivot table render module. It holds following parameters like **dataSourceSettings**, **fieldsType** and **pivotView**.

### CSHTML



```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource
=> dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingbar(true).Load("load").Render()
<script>
 function load(args){
 args.dataSourceSettings.columns[0].caption = 'Full Year';
 }
</script>
```

**LOAD.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*EnginePopulated*

The event **EnginePopulated** is triggered after engine is populated. It has following parameters - **dataSourceSettings**, **pivotFieldList** and **pivotValues**.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource
=> dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingbar(true).EnginePopulated("enginePopulated").Render()
<script>
 function enginePopulated(args){
 //trigger after engine populated
 }
</script>
```

**ENGINEPOPULATED.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*EnginePopulating*

The event **EnginePopulating** triggers before the pivot engine starts to populate and allows to customize the pivot datasource settings. It has following parameter **dataSourceSettings**.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingbar(true).EnginePopulating("enginePopulating").Render()
<script>
 function enginePopulating(args) {
 args.dataSourceSettings.columns[0].caption = 'Full Year';
 }
</script>
```

**ENGINEPOPULATING.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

## See Also

- [Aggregation](#)
- [Show/Hide Totals](#)
- [Customize number, date, and time values](#)
- [Server Side Engine \(Optional\)](#)

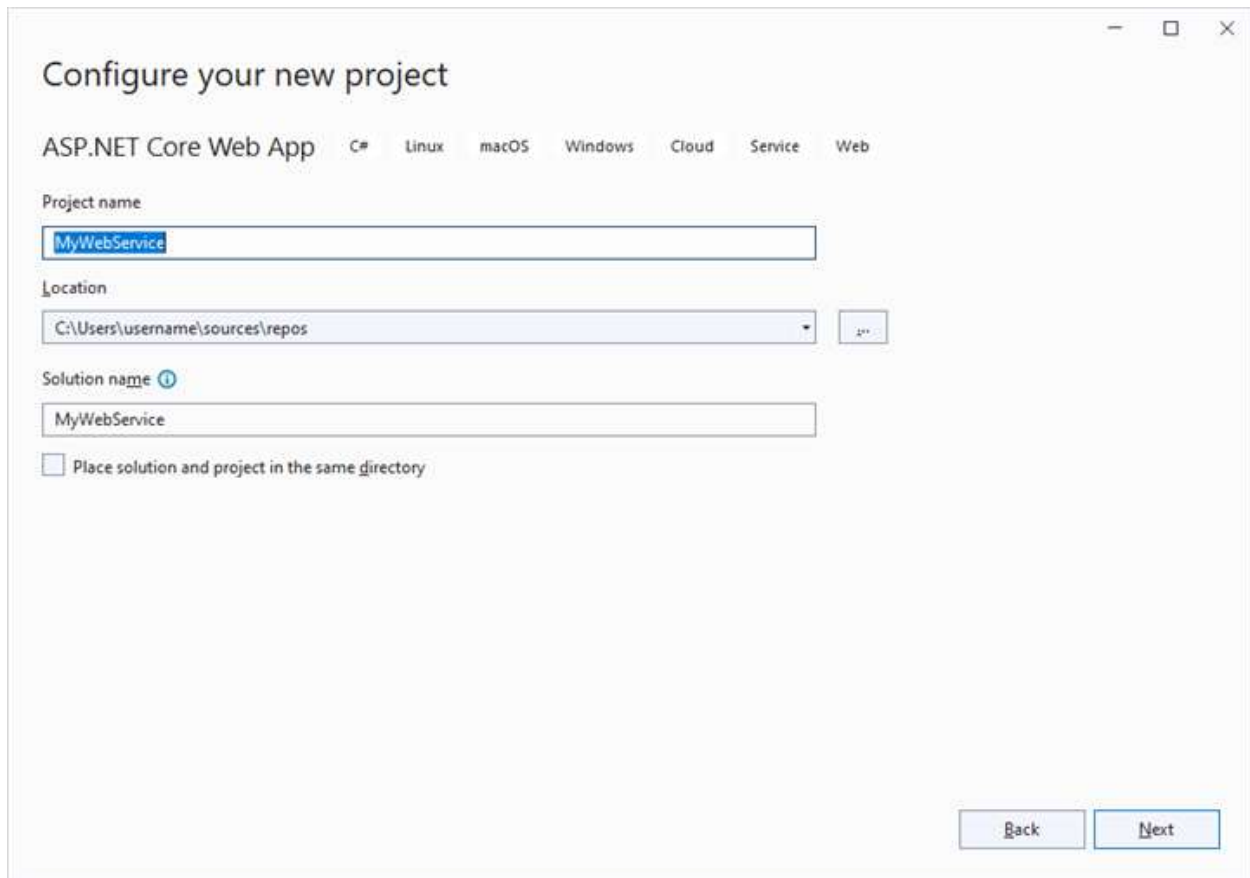
## Connecting to data source

### MySQL in EJ2 ASP.NET MVC Pivotview Component

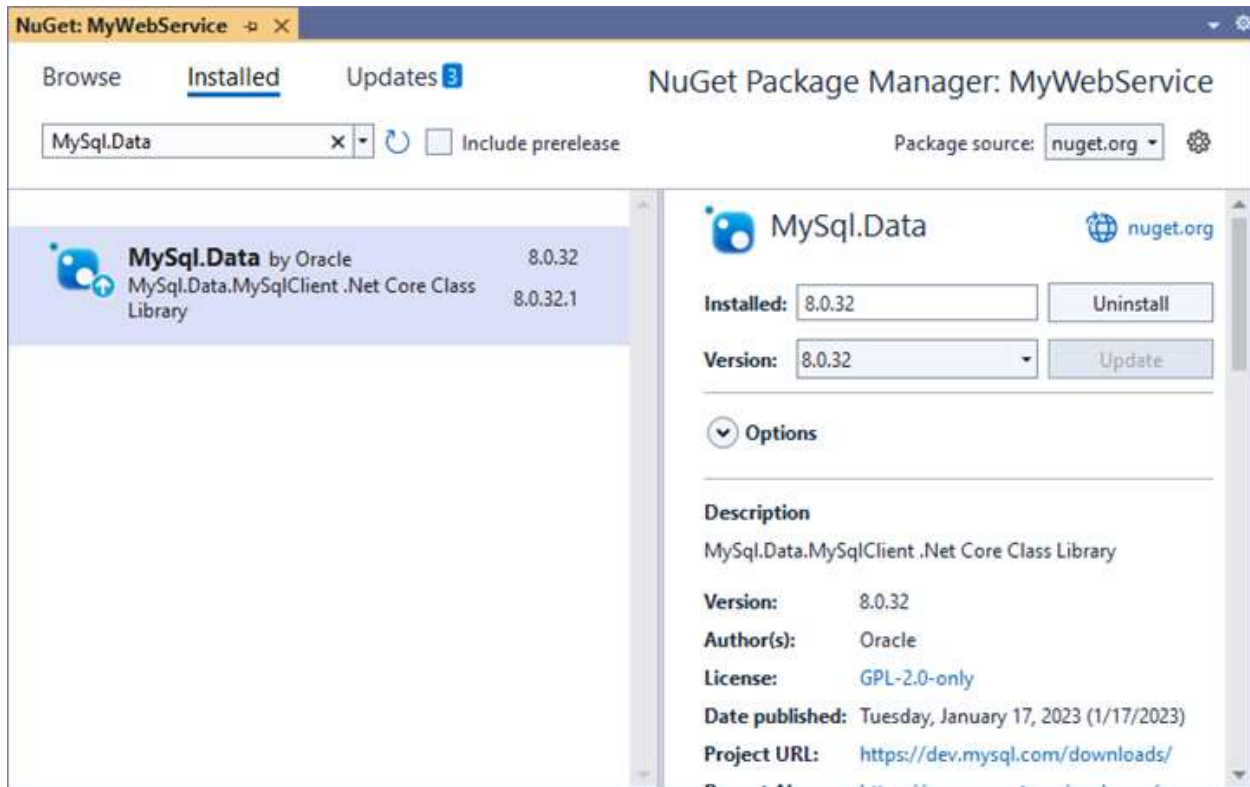
This section describes how to retrieve data from a MySQL database using [MySqlConnection](#) and bind it to the Pivot Table via a Web API controller.

#### *Create a Web API service to fetch MySQL data*

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



2. To connect a MySQL Server using the **MySqlConnection** in our application, we need to install the [MySQL.Data](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **MySQL.Data** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **MySQLConnection** helps to connect the MySQL database. Next, using **MySQLCommand** and **MySQLDataAdapter** you can process the desired query string and retrieve data from the MySQL database. The **Fill** method of the **MySQLDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using MySQL.Data.MySqlClient;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 public dynamic GetMySQLResult()
 {
```

```
// Replace with your own connection string.
MySQLConnection connection = new MySqlConnection("<Enter your valid connection string here>");
connection.Open();
MySQLCommand command = new MySQLCommand("SELECT * FROM orders", connection);
MySQLDataAdapter dataAdapter = new MySQLDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
connection.Close();
return dataTable;
}
}
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **GetMySQLResult** method is used to retrieve the MySQL data as a **DataTable**, which is then serialized into JSON string using **JsonConvert.SerializeObject()**.

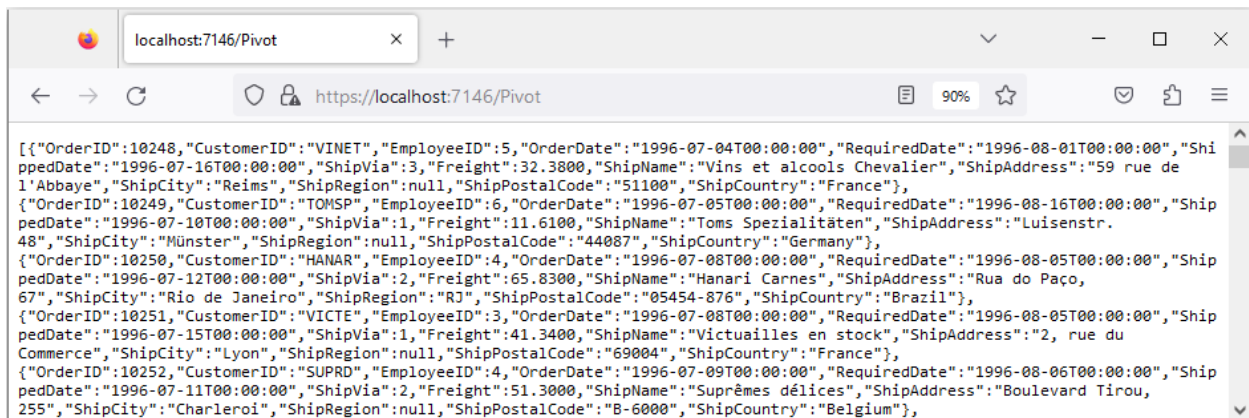
```
`csharp
using Microsoft.AspNetCore.Mvc;
using MySql.Data.MySqlClient;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetMySQLResult")]
 public object Get()
 {
 return JsonConvert.SerializeObject(GetMySQLResult());
 }
 public dynamic GetMySQLResult()
 {

```

```
// Replace with your own connection string.
MySQLConnection connection = new MySqlConnection("<Enter your valid connection string here>");
connection.Open();
MySQLCommand command = new MySQLCommand("SELECT * FROM orders", connection);
MySQLDataAdapter dataAdapter = new MySQLDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
connection.Close();
return dataTable;
}
}
}
```

6. Run the application and it will be hosted within the URL <https://localhost:7146>.

7. Finally, the retrieved data from MySQL database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:7146/Pivot>, as shown in the browser page below.



### Connecting the Pivot Table to a MySQL database using the Web API service

1. Create a simple ASP.NET MVC Pivot Table by following the **“Getting Started”** documentation [link](#).

2. Map the hosted Web API's URL link <https://localhost:7146/Pivot> to the Pivot Table in `~/Views/Home/Index.cshtml` by using the [Url](#) under [PivotViewDataSourceSettings](#) property.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:7146/pivot")
)
//Other codes here...
```

```

).Render()

```

3. Frame and set the report based on the data retrieved from the MySQL database.

```

`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:7146/Pivot"
).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
rows.Name("ShipCity").Add();
}).Columns(columns =>
{
columns.Name("ShipName").Add();
}).Values(values =>
{
values.Name("Freight").Caption("Sum of Freight").Add();
})).ShowFieldList(true).Render()

```

When you run the sample, the resulting pivot table will look like this:

|              | Alfred's Futterkist | Alfreds Futterkist | Ana Trujillo Empa | Antonio Moreno | Around the Horn | B's |
|--------------|---------------------|--------------------|-------------------|----------------|-----------------|-----|
| Aachen       |                     |                    |                   |                |                 |     |
| Albuquerque  |                     |                    |                   |                |                 |     |
| Anchorage    |                     |                    |                   |                |                 |     |
| Barcelona    |                     |                    |                   |                |                 |     |
| Barquisimeto |                     |                    |                   |                |                 |     |
| Bergamo      |                     |                    |                   |                |                 |     |
| Berlin       | 196.12000000...     | 29.46              |                   |                |                 |     |
| Bern         |                     |                    |                   |                |                 |     |

Explore our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to extract data from a MySQL database and bind to the Pivot Table in [this](#) GitHub repository.

Microsoft SQL Server in EJ2 ASP.NET MVC Pivotview Component

This section describes how to retrieve data from SQL Server database using [Microsoft SqlClient](#) and bind it to the Pivot Table via a Web API controller.

*Steps to connect the SQL Server database via Web API application*

1. Download the ASP.NET Core Web Application from [this](#) GitHub repository.

2. The application named as **PivotController** (server-side) that is downloaded from the above GitHub repository includes the following files.

- **PivotController.cs** file under **Controllers** folder – This helps to do data communication with Pivot Table.
- **Database1.mdf** file under **App\_Data** folder – This MDF (Master Database File) file contains example data.

3. In the Web API controller (aka, PivotController), **SqlConnection** helps to connect the SQL database (that is, Database1.mdf). Next, using **SqlCommand** and **SqlDataAdapter** you can process the desired SQL query string and retrieve data from the database. The **Fill** method of the DataAdapter is used to populate the SQL data into a **DataTable** as shown in the following code snippet.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using System.Data;
using System.Data.SqlClient;
namespace PivotController.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 private static DataTable FetchSQLResult()
 {
 string conSTR = @"<Enter your valid connection string here>";
 string xquery = "SELECT * FROM table1";
 SqlConnection sqlConnection = new(conSTR);
 sqlConnection.Open();
 SqlCommand cmd = new(xquery, sqlConnection);
 SqlDataAdapter dataAdapter = new(cmd);
 DataTable dataTable = new();
 dataAdapter.Fill(dataTable);
 return dataTable;
 }
 }
}
```



```
}
,
```

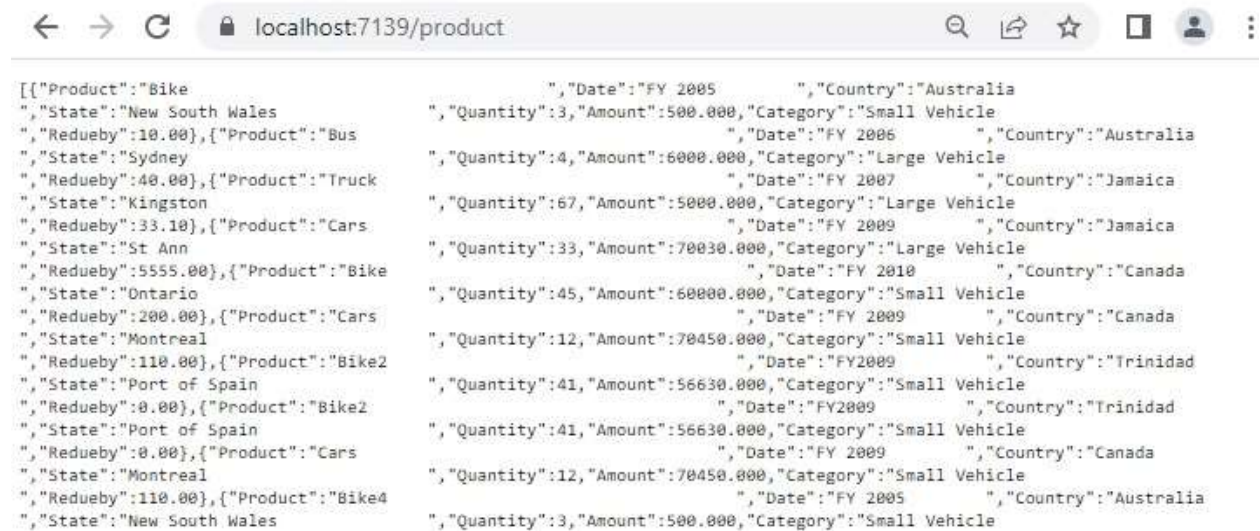
4. In the **Get()** method of the **PivotController.cs** file, the **FetchSQLResult** method is used to retrieve the SQL data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using System.Data.SqlClient;
namespace PivotController.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetSQLResult")]
 public object Get()
 {
 return JsonConvert.SerializeObject(FetchSQLResult());
 }
 private static DataTable FetchSQLResult()
 {
 string conSTR = @"<Enter your valid connection string here>";
 string xquery = "SELECT * FROM table1";
 SqlConnection sqlConnection = new(conSTR);
 sqlConnection.Open();
 SqlCommand cmd = new(xquery, sqlConnection);
 SqlDataAdapter dataAdapter = new(cmd);
 DataTable dataTable = new();
 dataAdapter.Fill(dataTable);
 return dataTable;
 }
 }
}
```

```
}
,
```

5. Run the web application (aka, PivotController) and it will be hosted within the URL <https://localhost:7139>.

6. Finally, the retrieved data from SQL Server which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:7139/pivot>, as shown in the browser page below.



#### Connecting the Pivot Table to the hosted Web API URL

1. Download the ASP.NET MVC Pivot Table sample from [this](#) GitHub repository.

2. Next, map the hosted Web API's URL link <https://localhost:7139/pivot> to the Pivot Table component in **app.ts** by using the [Url](#) under [PivotViewDataSourceSettings](#) property..

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:7139/pivot")
)
//Other codes here...
).Render()
,
```

3. Frame and set the report based on the data retrieved from the SQL database.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:7139/pivot")
).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
```

```

rows.Name("Country").Add(); rows.Name("State").Add();
}).Columns(columns =>
{
columns.Name("Product").Add();
}).Values(values =>
{
values.Name("Quantity").Add(); values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Render()
`

```

4. Run the sample to get the following result.

|             | Large Vehicle |        | Small Vehicle |         | Grand Total |    |
|-------------|---------------|--------|---------------|---------|-------------|----|
|             | Quantity      | Amount | Quantity      | Amount  | Quantity    | Am |
| ▶ Australia | 16            | 24000  | 12            | 2000    | 28          |    |
| ▶ Canada    |               |        | 420           | 1649000 | 420         |    |
| ▶ Jamaica   | 400           | 300120 |               |         | 400         |    |
| ▶ Trinidad  |               |        | 328           | 453040  | 328         |    |
| Grand Total | 416           | 324120 | 760           | 2104040 | 1176        |    |

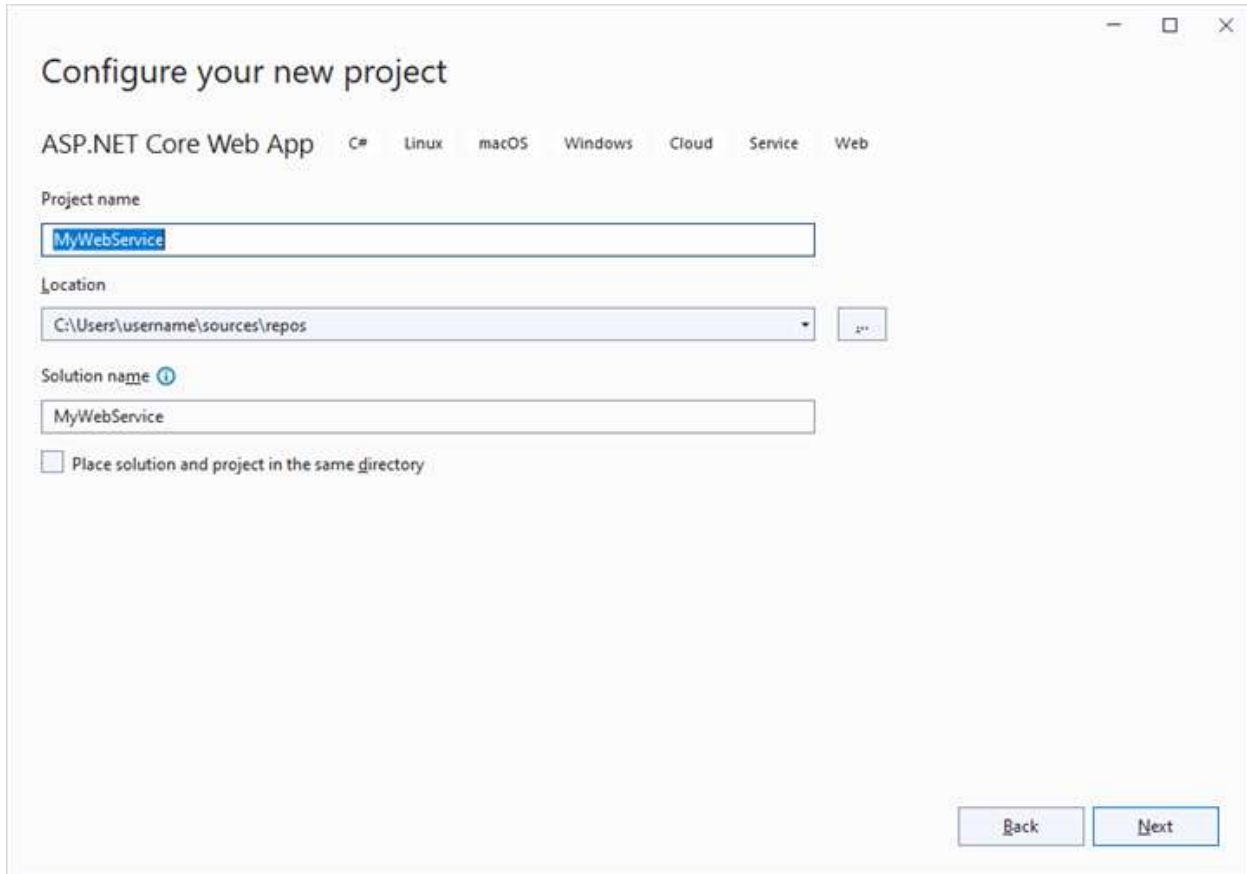
The sample for connecting the Pivot Table to a SQL Server database via an ASP.NET Web application can be found in [this](#) GitHub repository.

#### PostgreSQL in EJ2 ASP.NET MVC Pivotview Component

This section describes how to consume data from PostgreSQL database using [Microsoft Npgsql](#) and bind it to the Pivot Table via a Web API controller.

##### Create a Web API service to fetch PostgreSQL data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name  
MyWebService

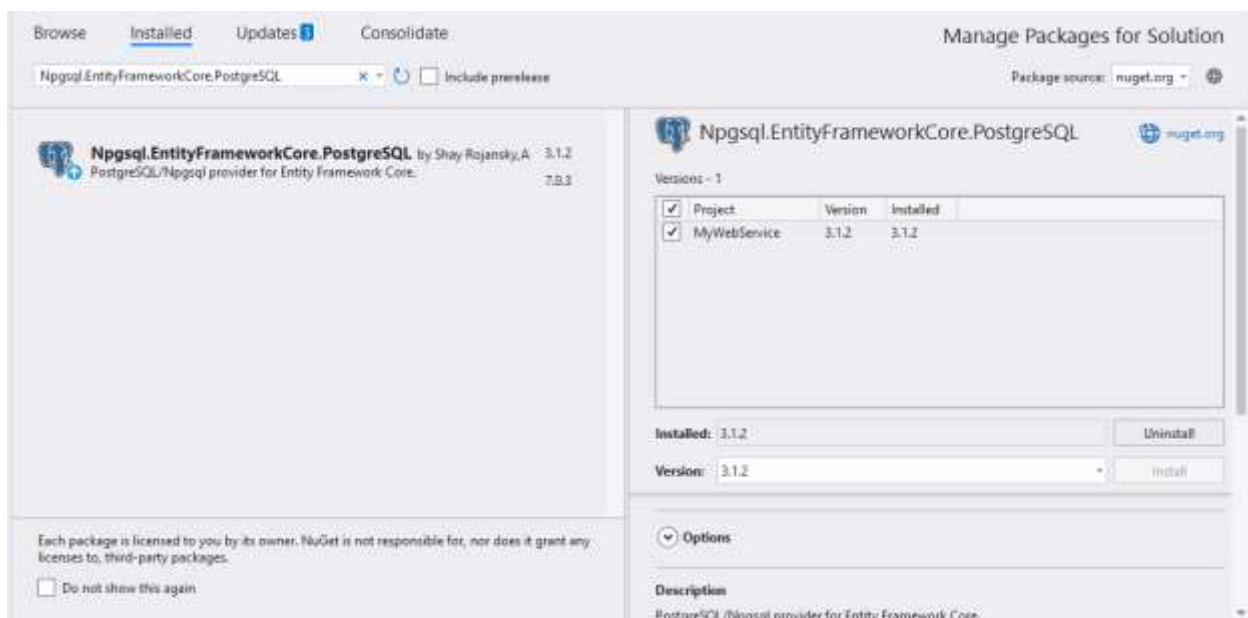
Location  
C:\Users\username\source\repos

Solution name  
MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a PostgreSQL Server using the **Npgsql** in our application, we need to install the [Npgsql.EntityFrameworkCore.PostgreSQL](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Npgsql.EntityFrameworkCore.PostgreSQL** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **NpgsqlConnection** helps to connect the PostgreSQL database. Next, using **NpgsqlCommand** and **NpgsqlDataAdapter** you can process the desired PostgreSQL query string and retrieve data from the database. The **Fill** method of the **NpgsqlDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using Npgsql;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 public dynamic GetPostgreSQLResult()
 {
 // Replace with your own connection string.
 NpgsqlConnection connection = new NpgsqlConnection("<Enter your valid connection string here>");
 connection.Open();
 NpgsqlCommand cmd = new NpgsqlCommand("SELECT * FROM tablename", connection);
 NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);
 DataTable dt = new DataTable();
 da.Fill(dt);
 connection.Close();
 return dt;
 }
 }
}
```

5. In the **Get()** method of the **PivotController.cs** file, the **GetPostgreSQLResult** method is used to retrieve the PostgreSQL data as a **DataTable**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`csharp
```

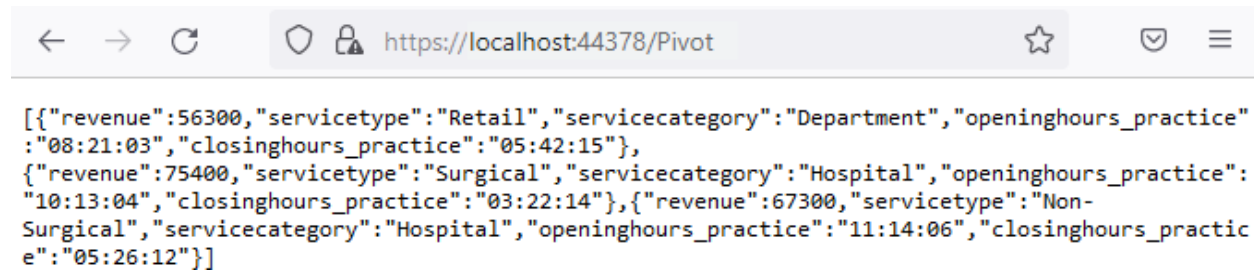
```

using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using System.Data;
using Npgsql;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetPostgreSQLResult")]
 public object Get()
 {
 return JsonConvert.SerializeObject(GetPostgreSQLResult());
 }
 public dynamic GetPostgreSQLResult()
 {
 // Replace with your own connection string.
 NpgsqlConnection connection = new NpgsqlConnection("<Enter your valid connection string here>");
 connection.Open();
 NpgsqlCommand cmd = new NpgsqlCommand("SELECT * FROM tablename", connection);
 NpgsqlDataAdapter da = new NpgsqlDataAdapter(cmd);
 DataTable dt = new DataTable();
 da.Fill(dt);
 connection.Close();
 return dt;
 }
 }
}

```

**6.** Run the application and it will be hosted within the URL <https://localhost:44378/>.

**7.** Finally, the retrieved data from PostgreSQL database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44378/Pivot>, as shown in the browser page below.



*Connecting the Pivot Table to a PostgreSQL database using the Web API service*

1. Create a simple ASP.NET MVC Pivot Table by following the **“Getting Started”** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44378/Pivot` to the Pivot Table component in `~/Views/Home/Index.cshtml` by using the [Url](#) under [PivotViewDataSourceSettings](#) property.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:44378/Pivot"
)
//Other codes here...
).Render()
`
```

3. Frame and set the report based on the data retrieved from the PostgreSQL database.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:44378/Pivot"
).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
rows.Name("servicetype").Add(); rows.Name("servicecategory").Add();
}).Columns(columns =>
{
columns.Name("openinghourspractice").Add(); columns.Name("closinghourspractice").Add();
}).Values(values =>
{
values.Name("revenue").Add();
})).ShowFieldList(true).Render()
`
```

When you run the sample, the resulting pivot table will look like this:

|                | ▶ 08:21:03 | ▶ 10:13:04 | ▶ 11:14:06 | Grand Total |
|----------------|------------|------------|------------|-------------|
| ▶ Non-Surgical |            |            | 67300      | 67300       |
| ▶ Retail       | 56300      |            |            | 56300       |
| ▶ Surgical     |            | 75400      |            | 75400       |
| Grand Total    | 56300      | 75400      | 67300      | 199000      |

Explore our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to extract data from a PostgreSQL database and bind to the Pivot Table in [this](#) GitHub repository.

#### Oracle in EJ2 ASP.NET MVC Pivotview Component

This section describes how to retrieve data from Oracle database using [Oracle Managed Data Access](#) and bind it to the Pivot Table via a Web API controller.

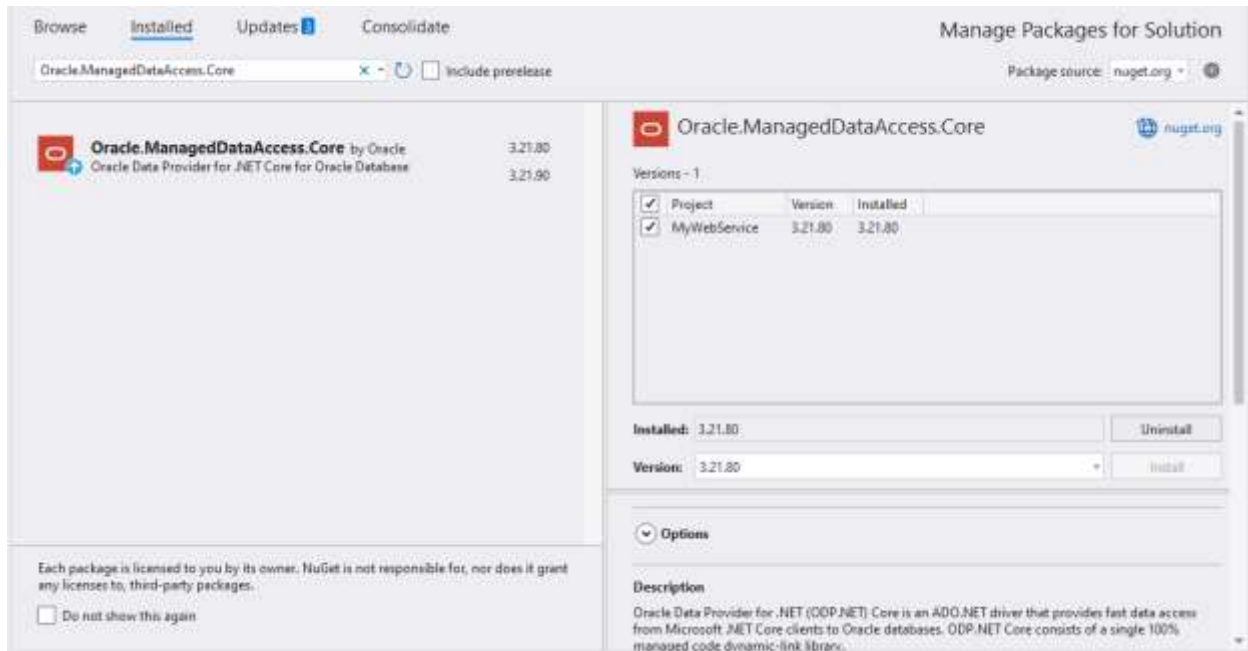
#### Create a Web API service to fetch Oracle data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

The screenshot shows the 'Configure your new project' window in Visual Studio. The 'ASP.NET Core Web App' template is selected. The 'Project name' field contains 'MyWebService'. The 'Location' field shows 'C:\Users\username\source\repos'. The 'Solution name' field also contains 'MyWebService'. There is an unchecked checkbox labeled 'Place solution and project in the same directory'. At the bottom right, there are 'Back' and 'Next' buttons.

2. To connect a Oracle Server using the **Oracle.ManagedDataAccess.Client** in our application, we need to install the [Oracle.ManagedDataAccess.Core](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Oracle.ManagedDataAccess.Core** and install it.





3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **OracleConnection** helps to connect the Oracle database. Next, using **OracleCommand** and **OracleDataAdapter** you can process the desired Oracle query string and retrieve data from the database. The **Fill** method of the **OracleDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Oracle.ManagedDataAccess.Client;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 private static DataTable FetchOracleResult()
 {
 // Replace with your own connection string.
 string connectionString = "<Enter your valid connection string here>";
 OracleConnection oracleConnection = new OracleConnection(connectionString);
```

```

oracleConnection.Open();
OracleCommand command = new OracleCommand("SELECT * FROM EMPLOYEES", oracleConnection);
OracleDataAdapter dataAdapter = new OracleDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
oracleConnection.Close();
return dataTable;
}
}
}
`

```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchOracleResult()** method is used to retrieve the Oracle data, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```

`csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using Oracle.ManagedDataAccess.Client;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetOracleResult")]
 public object Get()
 {
 return JsonConvert.SerializeObject(FetchOracleResult());
 }
 private static DataTable FetchOracleResult()
 {
 // Replace with your own connection string.
 string connectionString = "<Enter your valid connection string here>";

```

```

OracleConnection oracleConnection = new OracleConnection(connectionString);
oracleConnection.Open();
OracleCommand command = new OracleCommand("SELECT * FROM EMPLOYEES", oracleConnection);
OracleDataAdapter dataAdapter = new OracleDataAdapter(command);
DataTable dataTable = new DataTable();
dataAdapter.Fill(dataTable);
oracleConnection.Close();
return dataTable;
}
}
}
`

```

6. Run the application and it will be hosted within the URL <https://localhost:44346/>.

7. Finally, the retrieved data from Oracle database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44346/Pivot>, as shown in the browser page below.



*Connecting the Pivot Table to a Oracle database using the Web API service*

1. Create a simple ASP.NET MVC Pivot Table by following the “**Getting Started**” documentation [link](#).

2. Map the hosted Web API's URL link <https://localhost:44346/pivot> to the Pivot Table component in `~/Views/Home/Index.cshtml` by using the [Url](#) under [PivotViewDataSourceSettings](#) property.

```

`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:44346/pivot"
)
//Other codes here...

```

```

).Render()

```

3. Frame and set the report based on the data retrieved from the Oracle database.

```

`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:44346/Pivot"
).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
rows.Name("JOB").Caption("Job").Add(); rows.Name("SALARY").Caption("Salary").Add();
}).Columns(columns =>
{
columns.Name("DEPARTMENTID").Caption("Department ID").Add();
columns.Name("EMPLOYEEID").Caption("Employee ID").Add();
}).Values(values =>
{
values.Name("EMPLOYEEID").Caption("Employee ID").Add();
values.Name("CCEMPLOYEES").Caption("Employees").Add();
values.Name("CCTAXPERCENTAGE").Caption("Percentage").Add();
})).ShowFieldList(true).Render()

```

When you run the sample, the resulting pivot table will look like this:

|             | ▶ 10  | ▶ 20  | ▶ 30  | Grand Total |
|-------------|-------|-------|-------|-------------|
| ▶ ANALYST   |       | 15690 |       | 15690       |
| ▶ CLERK     | 7934  | 15245 | 7900  | 31079       |
| ▶ MANAGER   | 7782  | 7566  | 7698  | 23046       |
| ▶ PRESIDENT | 7839  |       |       | 7839        |
| ▶ SALESMAN  |       |       | 30518 | 30518       |
| Grand Total | 23555 | 38501 | 46116 | 108172      |

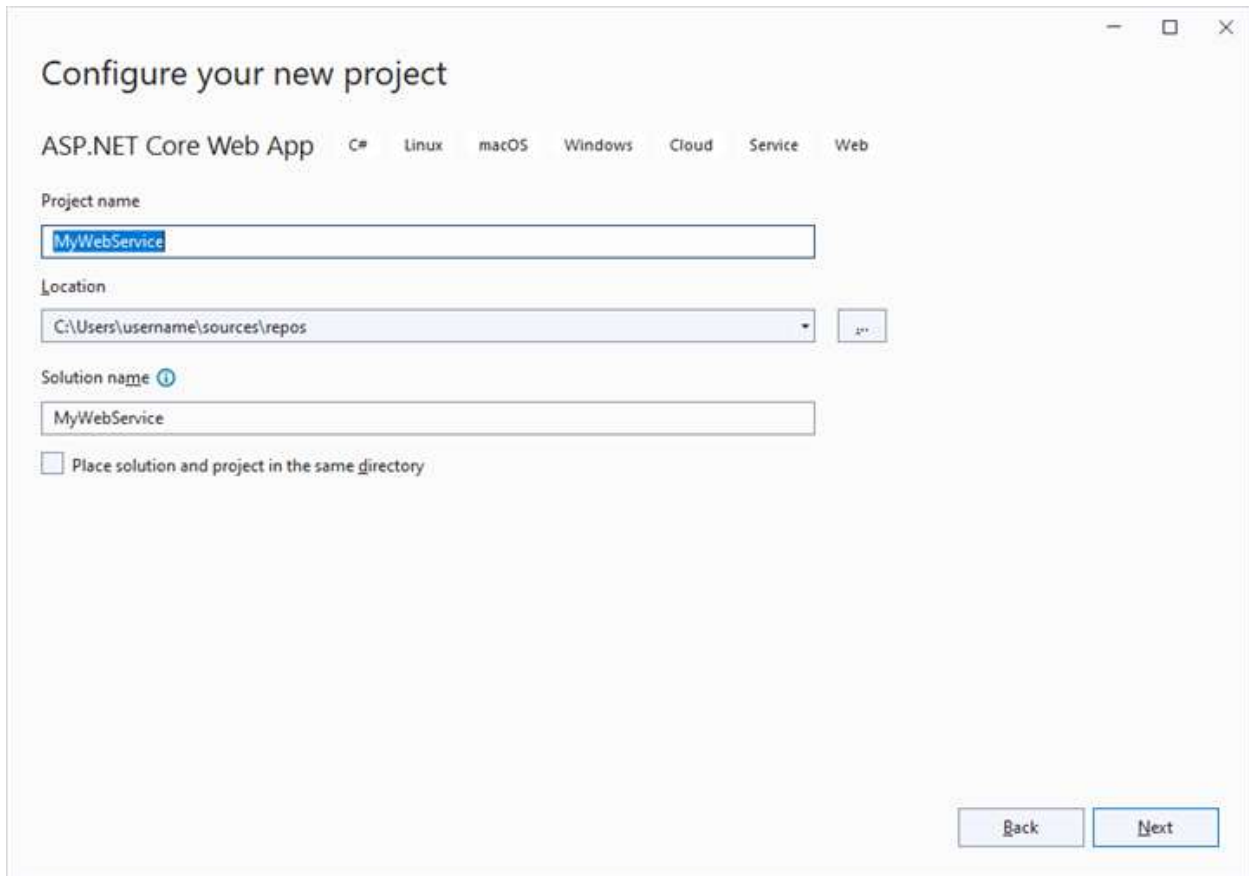
Explore our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to extract data from a Oracle database and bind to the Pivot Table in [this](#) GitHub repository.

#### MongoDB in EJ2 ASP.NET MVC Pivotview Component

This section describes how to consume data from MongoDB database using [MongoDB Driver](#) and [MongoDB Bson](#) libraries and bind it to the Pivot Table via a Web API controller.

*Create a Web API service to fetch MongoDB data*

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).



Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name

MyWebService

Location

C:\Users\username\source\repos

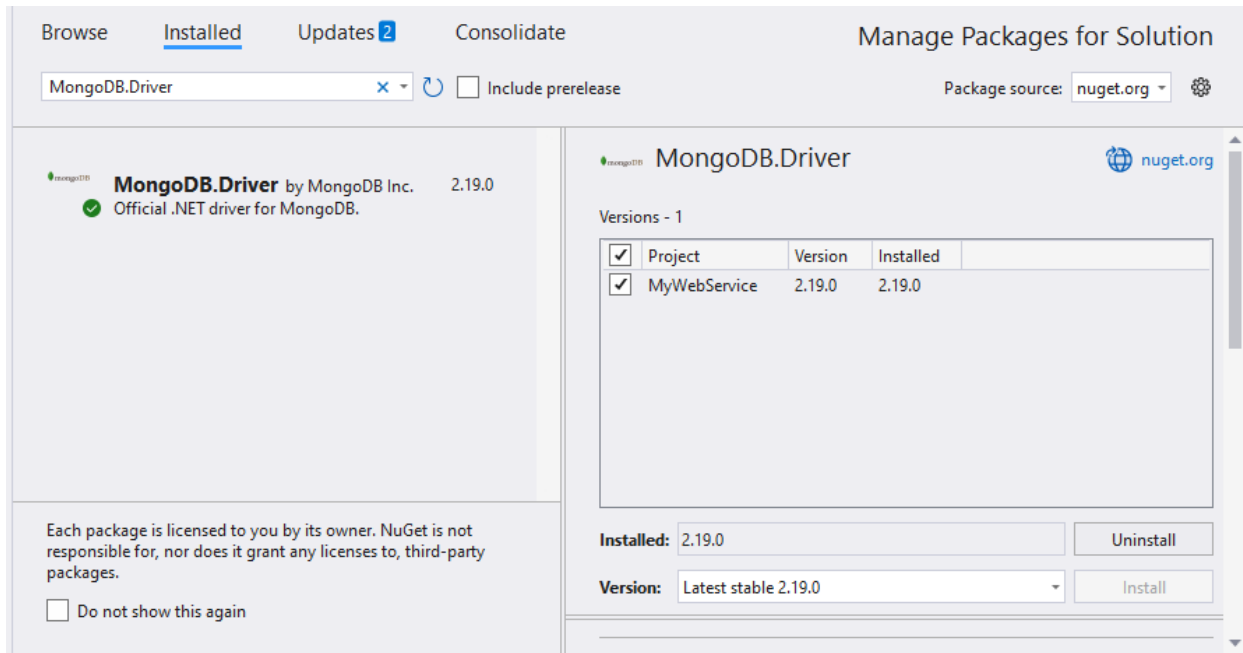
Solution name ⓘ

MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a MongoDB Server using the **MongoDB.Driver** and **MongoDB.Bson** in our application, we need to install the [MongoDB.Driver](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **MongoDB.Driver** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **MongoClient** helps to connect the MongoDB database. Next, using the **GetDatabase** and **GetCollection** methods, you can retrieve data from the database. The **Find** method of the **IMongoDatabase** is used to populate the retrieved data into a **List**, as shown in the following code snippet.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using MongoDB.Driver;
using MongoDB.Bson;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 private static List<ProductDetails> FetchMongoDbResult()
 {
 // Replace with your own connection string.
 string connectionString = "<Enter your valid connection string here>";
```

```

MongoClient client = new MongoClient(connectionString);
IMongoDatabase database = client.GetDatabase("sample_training");
var collection = database.GetCollection<ProductDetails>("ProductDetails");
return collection.Find(new BsonDocument()).ToList();
}

public class ProductDetails
{
 public ObjectId Id { get; set; }
 public int Sold { get; set; }
 public double Amount { get; set; }
 public string? Country { get; set; }
 public string? Products { get; set; }
 public string? Year { get; set; }
 public string? Quarter { get; set; }
}
}
}
`

```

5. In the **Get()** method of the **PivotController.cs** file, the **FetchMongoDbResult()** method is used to retrieve the MongoDB data as a **List**, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```

`csharp
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using MongoDB.Bson;
using MongoDB.Driver;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetMongoDbResult")]
 public object Get()

```

```

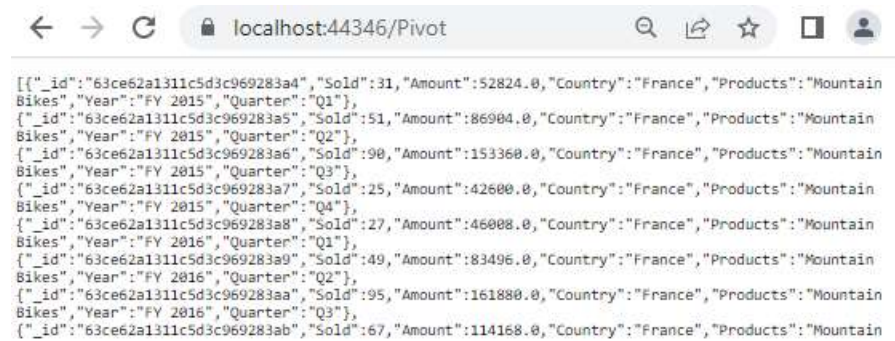
{
return JsonConvert.SerializeObject(FetchMongoDbResult());
}
private static List<ProductDetails> FetchMongoDbResult()
{
// Replace with your own connection string.
string connectionString = "<Enter your valid connection string here>";
MongoClient client = new MongoClient(connectionString);
IMongoDatabase database = client.GetDatabase("sample_training");
var collection = database.GetCollection<ProductDetails>("ProductDetails");
return collection.Find(new BsonDocument()).ToList();
}
public class ProductDetails
{
public ObjectId Id { get; set; }
public int Sold { get; set; }
public double Amount { get; set; }
public string? Country { get; set; }
public string? Products { get; set; }
public string? Year { get; set; }
public string? Quarter { get; set; }
}
}
}
`

```

6. Run the web application and it will be hosted within the URL <https://localhost:44346/>.

7. Finally, the retrieved data from MongoDB database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44346/Pivot>, as shown in the browser page below.





```
[{"_id": "63ce62a1311c5d3c969283a4", "Sold": 31, "Amount": 52824.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2015", "Quarter": "Q1"}, {"_id": "63ce62a1311c5d3c969283a5", "Sold": 51, "Amount": 86904.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2015", "Quarter": "Q2"}, {"_id": "63ce62a1311c5d3c969283a6", "Sold": 90, "Amount": 153360.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2015", "Quarter": "Q3"}, {"_id": "63ce62a1311c5d3c969283a7", "Sold": 25, "Amount": 42600.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2015", "Quarter": "Q4"}, {"_id": "63ce62a1311c5d3c969283a8", "Sold": 27, "Amount": 46008.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2016", "Quarter": "Q1"}, {"_id": "63ce62a1311c5d3c969283a9", "Sold": 49, "Amount": 83496.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2016", "Quarter": "Q2"}, {"_id": "63ce62a1311c5d3c969283aa", "Sold": 95, "Amount": 161880.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2016", "Quarter": "Q3"}, {"_id": "63ce62a1311c5d3c969283ab", "Sold": 67, "Amount": 114168.0, "Country": "France", "Products": "Mountain Bikes", "Year": "FY 2016", "Quarter": "Q4"}]
```

*Connecting the Pivot Table to a MongoDB database using the Web API service*

1. Create a simple ASP.NET MVC Pivot Table by following the **"Getting Started"** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44346/Pivot` to the Pivot Table component in `~/Views/Home/Index.cshtml` by using the [Url](#) under [PivotViewDataSourceSettings](#) property.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:44346/pivot")
)
//Other codes here...
).Render()
`
```

3. Frame and set the report based on the data retrieved from the MongoDB database.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:44346/Pivot")
).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add(); values.Name("Amount").Caption("Sold Amount").Add();
}).ShowFieldList(true).Render()
`
```

When you run the sample, the resulting pivot table will look like this:

|                  | FY 2015    |             | FY 2016    |             | F |
|------------------|------------|-------------|------------|-------------|---|
|                  | Units Sold | Sold Amount | Units Sold | Sold Amount |   |
| > France         | 729        | 1160099.5   | 609        | 983317      |   |
| > Germany        | 528        | 845472      | 667        | 1067220     |   |
| > United Kingdom | 782        | 1263109.5   | 640        | 1031630.5   |   |
| > United States  | 682        | 1085398.5   | 480        | 770362      |   |
| Grand Total      | 2721       | 4354079.5   | 2396       | 3852529.5   |   |

Explore our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to extract data from a MongoDB database and bind to the Pivot Table in [this](#) GitHub repository.

### Elasticsearch in EJ2 ASP.NET MVC Pivotview Component

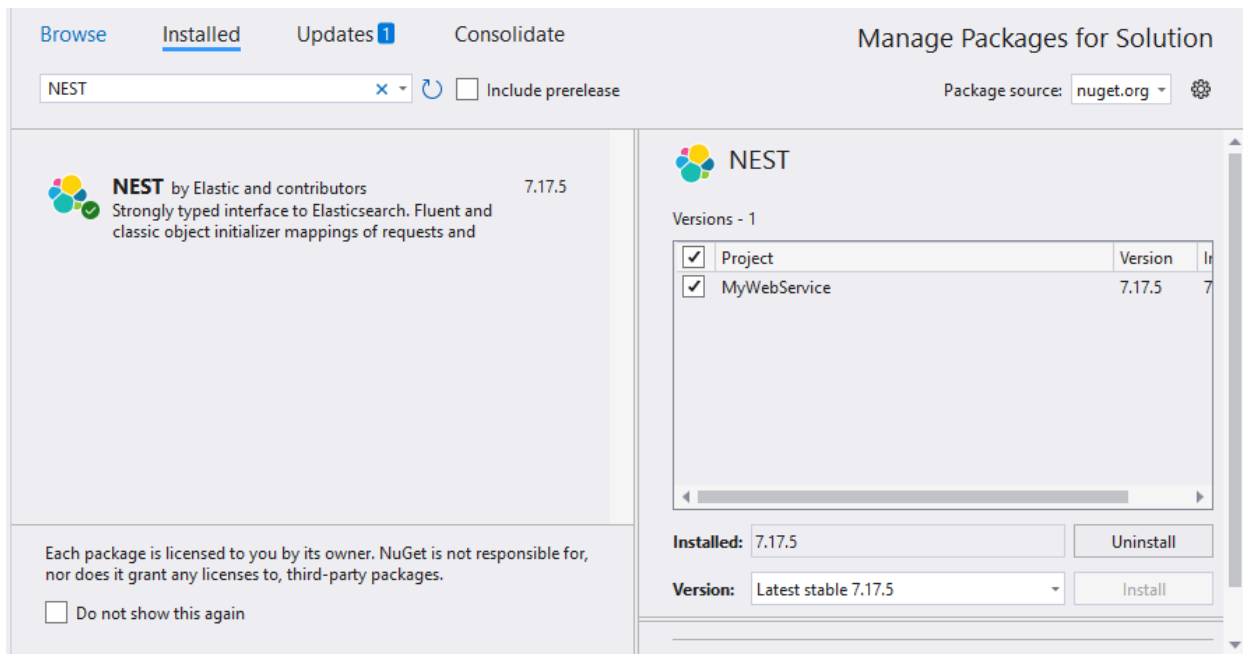
This section describes how to retrieve data from Elasticsearch database using [Nest](#) library and bind it to the Pivot Table via a Web API controller.

#### Create a Web API service to fetch Elasticsearch data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

The screenshot shows the 'Configure your new project' dialog in Visual Studio. The 'ASP.NET Core Web App' template is selected. The project name is 'MyWebService', the location is 'C:\Users\username\sources\repos', and the solution name is 'MyWebService'. The checkbox 'Place solution and project in the same directory' is unchecked. 'Back' and 'Next' buttons are at the bottom right.

2. To connect a Elasticsearch Server using the **NEST** in our application, we need to install the [NEST](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **NEST** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **ElasticClient** helps to connect the Elasticsearch database. Next, using **Search** method you can query your Elasticsearch index and retrieve results from the database.

5. In the **Get()** method of the **PivotController.cs** file, the **FetchElasticsearchData** method is used to retrieve the Elasticsearch data, which is then serialized into JSON using **JsonConvert.SerializeObject()**.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Nest;
using Newtonsoft.Json;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetElasticSearchData")]
 public object Get()
```

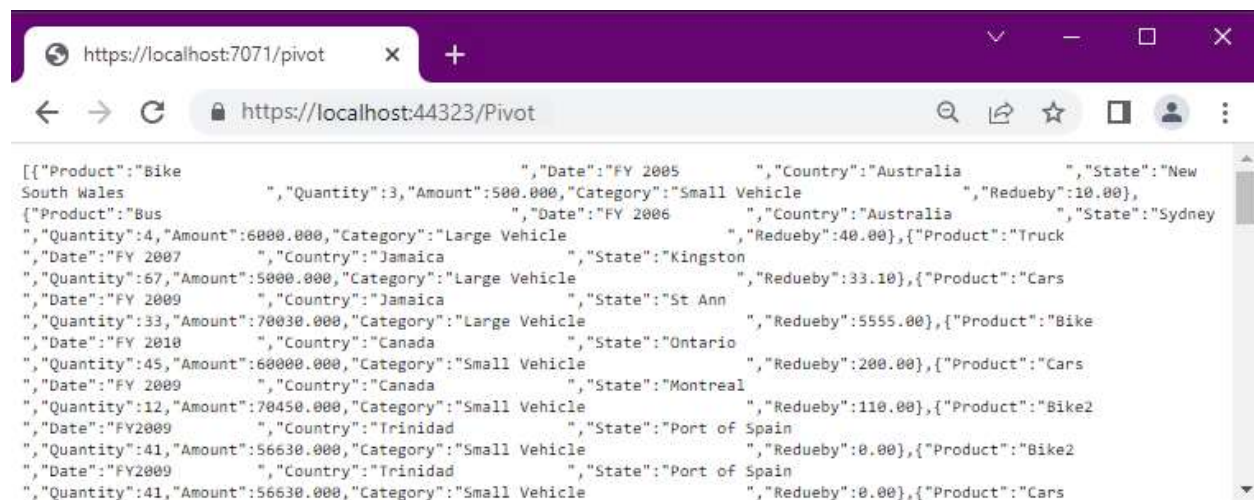
```

{
return JsonConvert.SerializeObject(FetchElasticsearchData());
}
private static object FetchElasticsearchData()
{
// Replace with your own connection string.
var connectionString = "<Enter your valid connection string here>";
var uri = new Uri(connectionString);
var connectionSettings = new ConnectionSettings(uri);
var client = new ElasticClient(connectionSettings);
var searchResponse = client.Search<object>(s => s
.Index("product")
.Size(1000)
);
return searchResponse.Documents;
}
}
}
,

```

6. Run the web application and it will be hosted within the URL <https://localhost:44323>.

7. Finally, the retrieved data from Elasticsearch database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44323/Pivot>, as shown in the browser page below.



*Connecting the Pivot Table to a Elasticsearch database using the Web API service*

1. Create a simple ASP.NET MVC Pivot Table by following the **"Getting Started"** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44323/Pivot` to the Pivot Table component in `~/Views/Home/Index.cshtml` by using the [Url](#) under [PivotViewDataSourceSettings](#) property.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:44323/pivot"
)
//Other codes here...
).Render()
`
```

3. Frame and set the report based on the data retrieved from the Elasticsearch database.

```
`csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
 dataSource => dataSource.Url("https://localhost:44323/Pivot"
).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("State").Add();
}).Columns(columns =>
{
 columns.Name("Product").Add();
}).Values(values =>
{
 values.Name("Quantity").Add(); values.Name("Amount").Caption("Sold Amount").Add();
}).ShowFieldList(true).Render()
`
```

When you run the sample, the resulting pivot table will look like this:

|             | Large Vehicle |        | Small Vehicle |         | Grand Total |        |
|-------------|---------------|--------|---------------|---------|-------------|--------|
|             | Quantity      | Amount | Quantity      | Amount  | Quantity    | Amount |
| ▶ Australia | 16            | 24000  | 12            | 2000    | 28          |        |
| ▶ Canada    |               |        | 420           | 1649000 | 420         |        |
| ▶ Jamaica   | 400           | 300120 |               |         | 400         |        |
| ▶ Trinidad  |               |        | 328           | 453040  | 328         |        |
| Grand Total | 416           | 324120 | 760           | 2104040 | 1176        |        |

Explore our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to extract data from a Elasticsearch database and bind to the Pivot Table in [this](#) GitHub repository.

### Snowflake in EJ2 ASP.NET MVC Pivotview Component

This section describes how to retrieve data from a Snowflake database using [Snowflake Data](#) and bind it to the Pivot Table via a Web API controller.

#### Create a Web API service to fetch Snowflake data

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

Configure your new project

ASP.NET Core Web App   C#   Linux   macOS   Windows   Cloud   Service   Web

Project name  
MyWebService

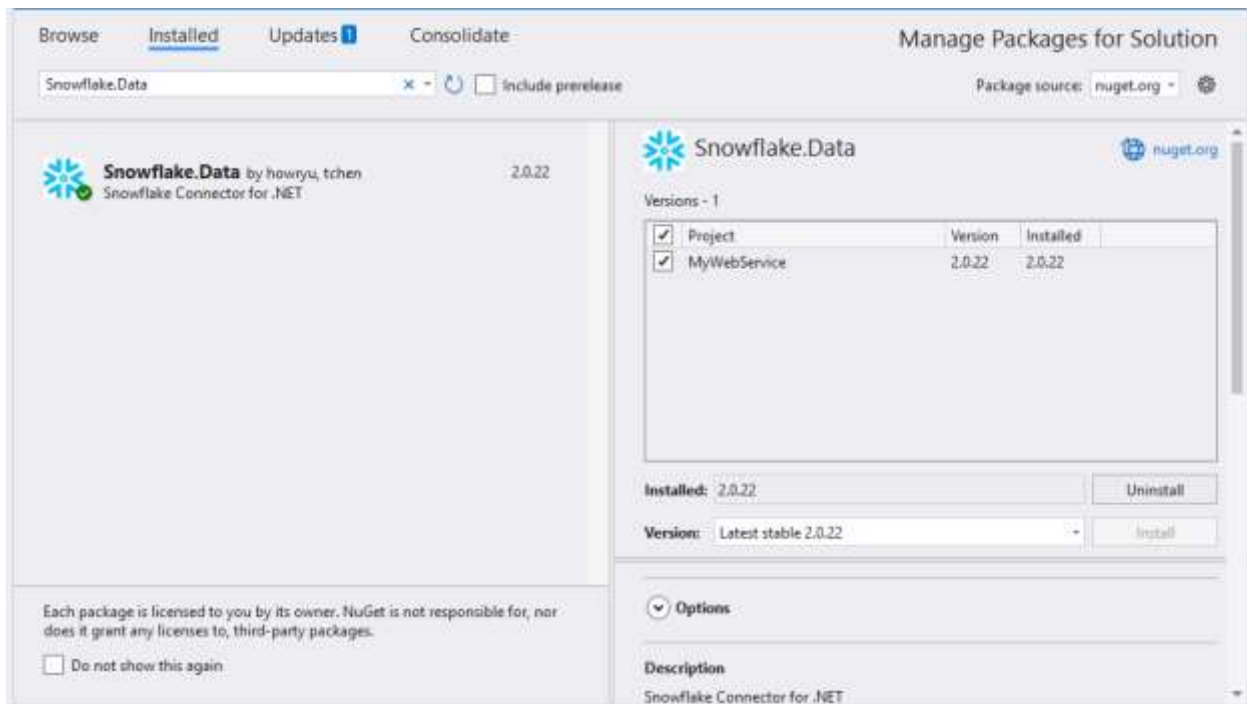
Location  
C:\Users\username\source\repos

Solution name ⓘ  
MyWebService

☐ Place solution and project in the same directory

Back   Next

2. To connect a Snowflake Server using the **Snowflake.Data.Client** in our application, we need to install the [Snowflake.Data](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Snowflake.Data** and install it.



3. Create a Web API controller (aka, PivotController.cs) file under **Controllers** folder that helps to establish data communication with the Pivot Table.

4. In the Web API controller (aka, PivotController), **SnowflakeDbConnection** helps to connect the Snowflake database. Next, using **SnowflakeDbDataAdapter** you can process the desired Snowflake query string and retrieve data from the database. The **Fill** method of the **SnowflakeDbDataAdapter** is used to populate the retrieved data into a **DataTable** as shown in the following code snippet.

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Snowflake.Data.Client;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpGet(Name = "GetSnowflakeResult")]
```

```

public object Get()
{
 return JsonConvert.SerializeObject(FetchSnowflakeResult());
}

public static DataTable FetchSnowflakeResult()
{
 using (SnowflakeDbConnection snowflakeConnection = new SnowflakeDbConnection())
 {
 // Replace with your own connection string.
 snowflakeConnection.ConnectionString = "<Enter your valid connection string here>";
 snowflakeConnection.Open();

 SnowflakeDbDataAdapter adapter = new SnowflakeDbDataAdapter("select * from CALL_CENTER",
 snowflakeConnection);

 DataTable dataTable = new DataTable();
 adapter.Fill(dataTable);
 snowflakeConnection.Close();
 return dataTable;
 }
}

```

**5.** In the **Get()** method of the **PivotController.cs** file, the **FetchSnowflakeResult** method is used to retrieve the Snowflake data as a **DataTable**, which is then serialized into JSON string using **JsonConvert.SerializeObject()**.

```

`csharp
using Microsoft.AspNetCore.Mvc;
using Snowflake.Data.Client;
using Newtonsoft.Json;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]

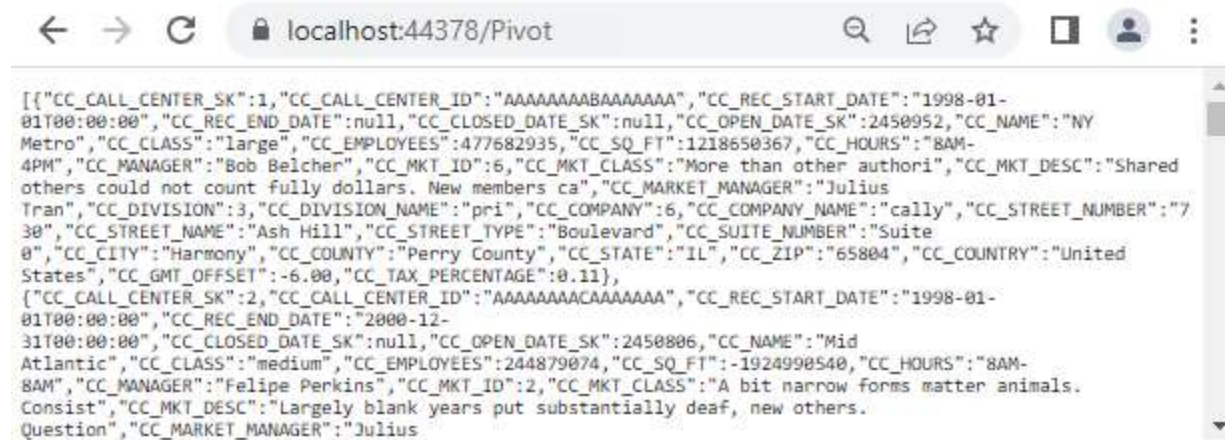
```



```
public class PivotController : ControllerBase
{
 [HttpGet(Name = "GetSnowflakeResult")]
 public object Get()
 {
 return JsonConvert.SerializeObject(FetchSnowflakeResult());
 }
 public static DataTable FetchSnowflakeResult()
 {
 using (SnowflakeDbConnection snowflakeConnection = new SnowflakeDbConnection())
 {
 // Replace with your own connection string.
 snowflakeConnection.ConnectionString = "<Enter your valid connection string here>";
 snowflakeConnection.Open();
 SnowflakeDbDataAdapter adapter = new SnowflakeDbDataAdapter("select * from CALL_CENTER",
 snowflakeConnection);
 DataTable dataTable = new DataTable();
 adapter.Fill(dataTable);
 snowflakeConnection.Close();
 return dataTable;
 }
 }
}
```

6. Run the application and it will be hosted within the URL <https://localhost:44378/>.

7. Finally, the retrieved data from Snowflake database which is in the form of JSON can be found in the Web API controller available in the URL link <https://localhost:44378/Pivot>, as shown in the browser page below.



*Connecting the Pivot Table to a Snowflake database using the Web API service*

1. Create a simple ASP.NET MVC Pivot Table by following the **"Getting Started"** documentation [link](#).
2. Map the hosted Web API's URL link `https://localhost:44378/Pivot` to the Pivot Table component in `~/Views/Home/Index.cshtml` by using the [Url](#) under [PivotViewDataSourceSettings](#) property.

```
'csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:44378/pivot"
)
//Other codes here...
).Render()
`
```

3. Frame and set the report based on the data retrieved from the Snowflake database.

```
'csharp
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(
dataSource => dataSource.Url("https://localhost:44378/Pivot"
).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
rows.Name("CCSTATE").Caption("State").Add(); rows.Name("CCCITY").Caption("City").Add();
}).Columns(columns =>
{
columns.Name("CC_COUNTRY").Caption("Country").Add();
}).Values(values =>
{

```

```

values.Name("CCCOMPANY").Caption("Company").Add();
values.Name("CCEMPLOYEES").Caption("Employees").Add();
values.Name("CCTAXPERCENTAGE").Caption("Percentage").Add();
}).ShowFieldList(true).Render()

```

When you run the sample, the resulting pivot table will look like this:

|      | United States |            |                 | Grand Total |            |             |
|------|---------------|------------|-----------------|-------------|------------|-------------|
|      | Company       | Employees  | Percentage      | Company     | Employees  | Percentage  |
| ▶ CA | 16            | 413279139  | 0.4800000000... | 16          | 413279139  | 0.480000... |
| ▶ CO | 5             | 1160903623 | 0.18            | 5           | 1160903623 |             |
| ▶ FL | 15            | 1162362540 | 0.33            | 15          | 1162362540 |             |
| ▶ GA | 20            | 2358562323 | 0.12            | 20          | 2358562323 |             |
| ▶ IL | 8             | 707852970  | 0.1699999999... | 8           | 707852970  | 0.169999... |
| ▶ KS | 9             | 1842349179 | 0.19            | 9           | 1842349179 |             |
| ▶ LA | 6             | 528468075  | 0.16            | 6           | 528468075  |             |

Explore our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to extract data from a Snowflake database and bind to the Pivot Table in [this](#) GitHub repository.

## OLAP

### Getting Started with ASP.NET MVC

**Note:** Starting with v16.2.0.x, if you reference Syncfusion assemblies from trial setup or from the NuGet feed, you also have to include a license key in your projects. Refer to this [link](#) to know about registering Syncfusion license key in your ASP.NET MVC application to use our components.

### Prerequisites

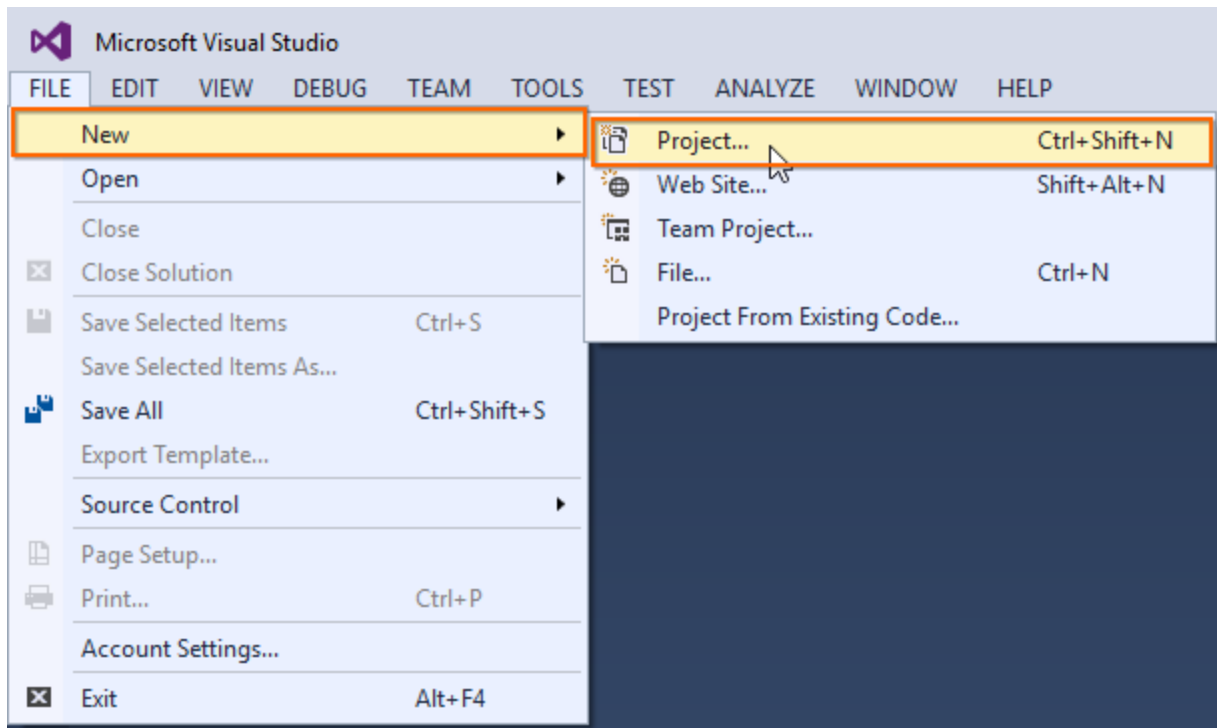
To get start with ASP.NET MVC application, need to ensure the following software to be installed on the machine.

1. .NET Framework 4.5 and above.
2. ASP.NET MVC 4 or ASP.NET MVC 5
3. Visual Studio

### Preparing ASP.NET MVC application

Follow below steps to create ASP.NET MVC Application.

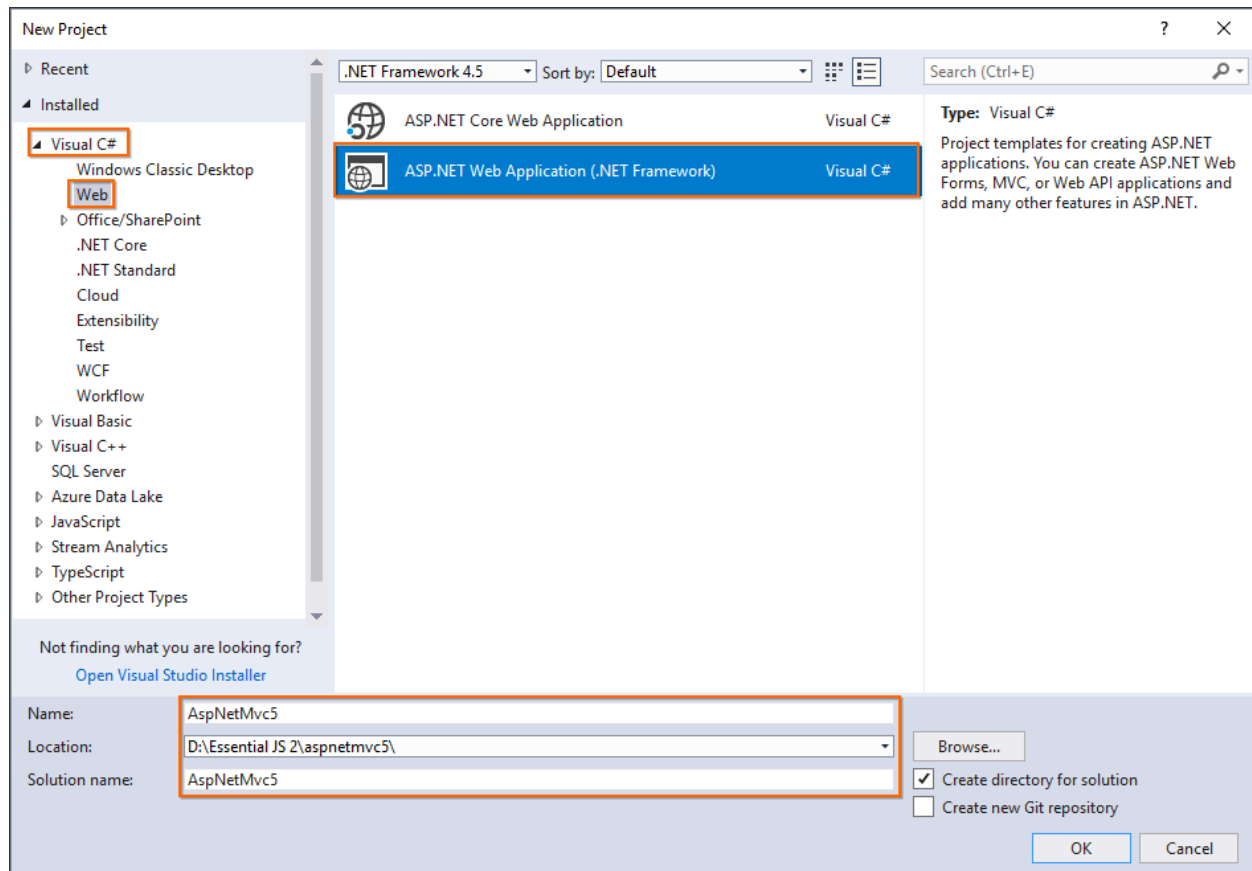
**Step 1:** Choose **File > New > Project...** in the Visual Studio menu bar.



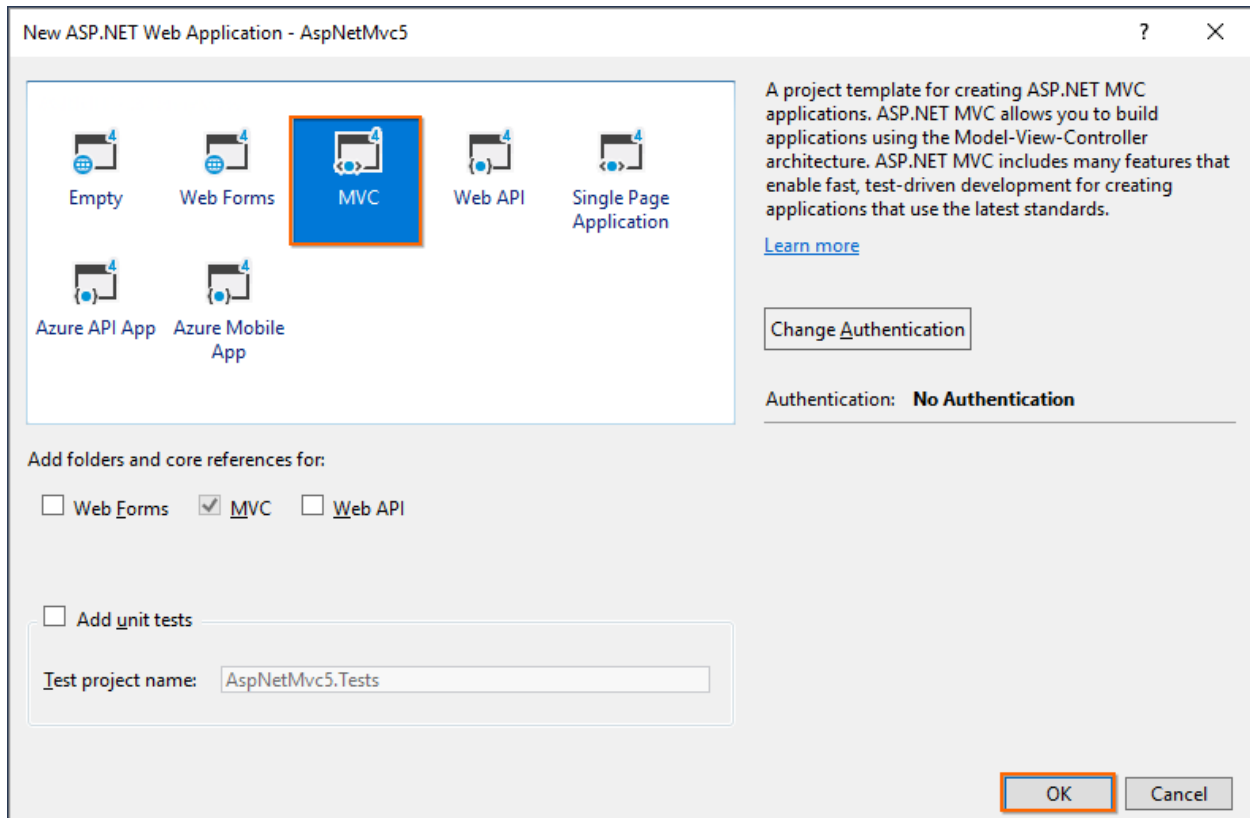
**Step 2:** Select **Installed > Visual C# > Web** and choose the required **.NET Framework** in the drop-down.

**Step 3:** Select **ASP.NET Web Application (.NET Framework)** and change the application name, and then click **OK**.

**Note:** The Essential JS 2 supports 4.5+ .NET Framework in the ASP.NET MVC application. i.e. The minimum target framework is 4.5 for Syncfusion ASP.NET MVC (Essential JS 2).



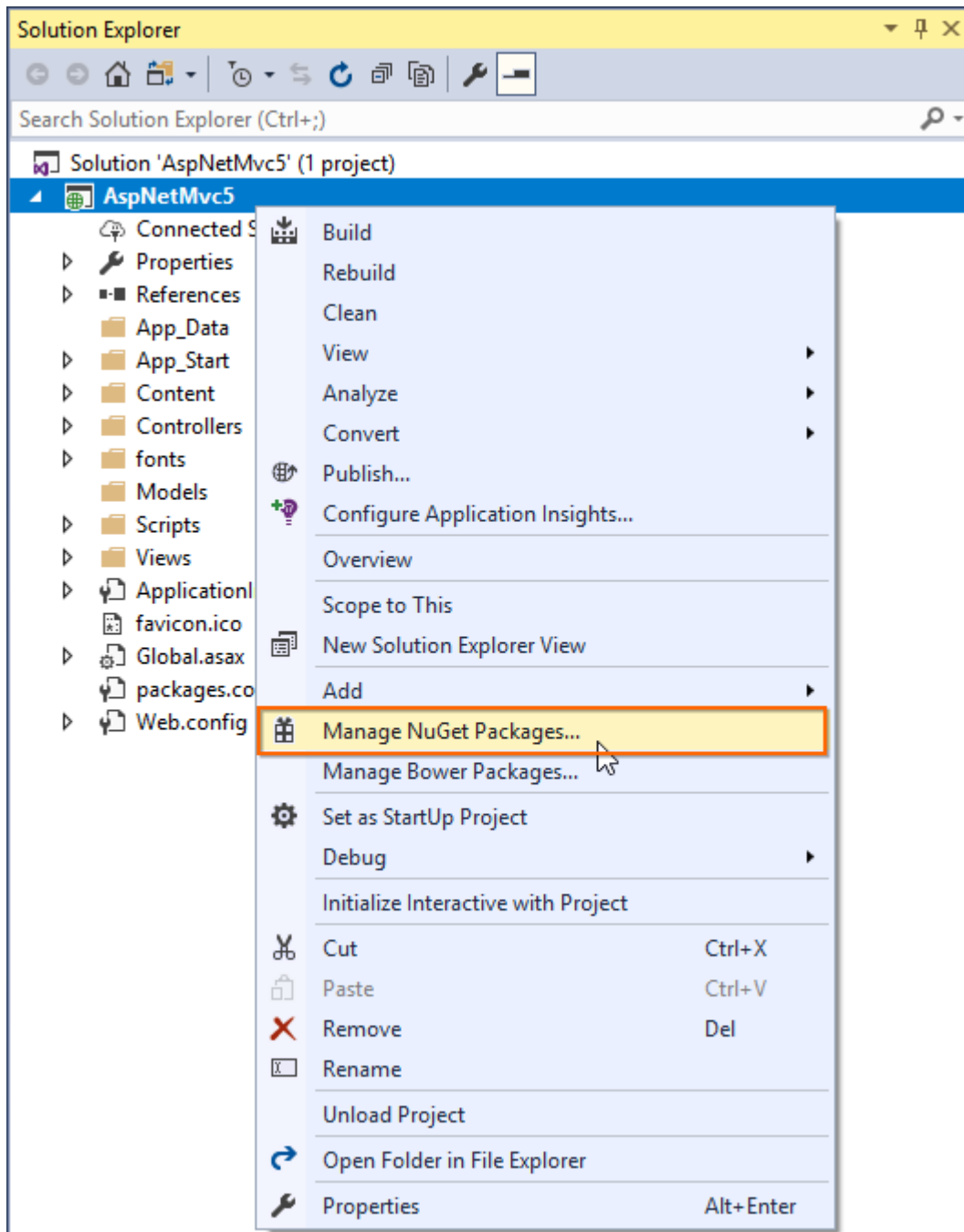
**Step 4:** Choose **MVC** and then click **OK**. Now, the MVC web application project is created with default ASP.NET MVC template.



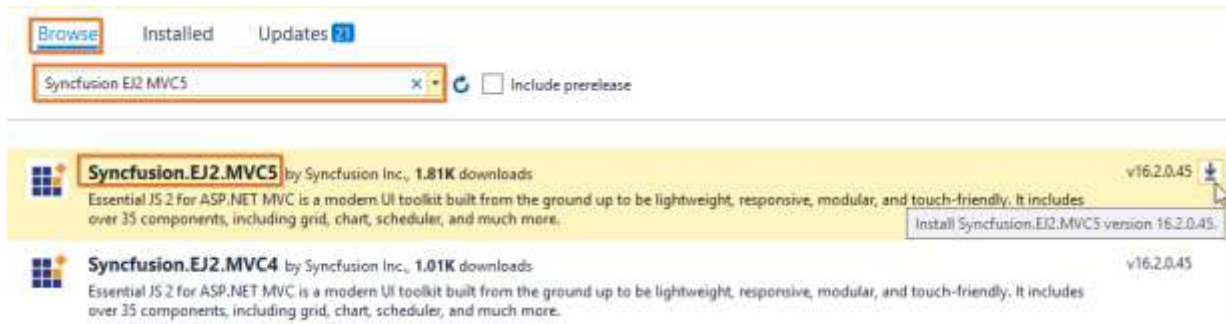
*Configure Essential JS 2 in the application*

**Step 1:** Add the [SynCFusion.EJ2.MVC5](#) NuGet package to the new application by using the Nuget Package Manager. Right-click the project and select **Manage NuGet Packages...**

**Note:** Refer to [this article](#) to learn more details about installing Essential JS 2 NuGet packages in various OS environment.



**Step 2:** Search the Syncfusion EJ2 MVC5 keyword in the **Browse** tab and install **Syncfusion.EJ2.MVC5** NuGet package in the application.



The Essential JS 2 MVC5 NuGet package will be included in the project, after the installation process is completed.

**Note:** The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Step 3:** Open `~/Views/Web.config` file and add the `Syncfusion.EJ2` namespace reference to the `<system.web.webPages.razor>` element and `Syncfusion.EJ2` assembly reference to `<system.web>` element.

```
`html
<configuration>
...
<system.web.webPages.razor>
...
<pages pageBaseType="System.Web.Mvc.WebViewPage">
<namespaces>
...
...
<add namespace="Syncfusion.EJ2"/>
</namespaces>
</pages>
</system.web.webPages.razor>
...
<system.web>
<compilation>
<assemblies>
...
...
<add assembly="Syncfusion.EJ2, Culture=neutral" />
</assemblies>
```



```

</compilation>
</system.web>
</configuration>
`

```

**Step 4:** Add the client-side resources through [CDN](#) in the `<head>` element of `~/Views/Shared/_Layout.cshtml` layout page.

```

`html
<head>
...
<!-- Syncfusion Essential JS 2 Styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/material.css" />
<!-- Syncfusion Essential JS 2 Scripts -->
<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
</head>
`

```

**Step 5:** Add the Essential JS 2 Script Manager at the end of `<body>` element in the `~/Views/Shared/_Layout.cshtml` layout page.

```

`html
<body>
...
<!-- Syncfusion Essential JS 2 ScriptManager -->
@Html.EJS().ScriptManager()
</body>
`

```

### Adding component to the application

Add the below code to your `Index.cshtml` view page which is present under `Views/Home` folder, to initialize the pivot table component with sample OLAP data source.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").DataSourceSettings(
dataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure
Works").ProviderType(ProviderType.SSAS)).Render()

```

### PIVOTTABLE.CS

```
public ActionResult Index()
{
 return View();
}
```

#### *Adding OLAP cube elements to row, column, value and filter axes*

Now that pivot table is initialized and assigned with sample OLAP data source, will further move to showcase the component by organizing appropriate [OLAP cube elements](#) in [Rows](#), [Columns](#), [Values](#) and [Filters](#) axes.

In [DataSourceSettings](#) property, four major axes [Rows](#), [Columns](#), [Values](#) and [Filters](#) plays a vital role in defining and organizing [OLAP cube elements](#) from the bound data source, to render the entire pivot table component in a desired format.

**Rows** – Collection of [OLAP cube elements](#) (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in row axis of the pivot table.

**Columns** – Collection of [OLAP cube elements](#) (such as Hierarchies, NamedSet, Calculated Members etc.,) that needs to be displayed in column axis of the pivot table.

**Values** – Collection of [OLAP cube elements](#) (such as Measures, Calculated Measures) that needs to be displayed as aggregated numeric values in the pivot table.

**Filters** - Collection of [OLAP cube elements](#) (such as Hierarchies and Calculated Members) that would act as master filter over the data bound in row, column and value axes of the pivot table.

In-order to define each [OLAP cube element](#) in the respective axis, the following basic properties should be set.

- **Name**: It allows to set the unique name of the hierarchies, named set, measures, calculated members etc., from the bound OLAP data source. It's casing should match exactly like in the data source and if not set properly, the pivot table will be rendered as empty.
- **Caption**: It allows to set the caption, which is the alias name of the unique name that needs to be displayed in the pivot table. If not provided, unique name will be displayed.

In this sample, "Product Categories" is added in column, "Customer Geography" in row, and "Customer Count" and "Internet Sales Amount" in value axes respectively.

#### **CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("400").DataSourceSettings(dataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product Categories]").Caption("Product Categories").Add(); columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet Sales Amount]").Caption("Internet Sales Amount").Add(); })
```

```
.Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date Fiscal").Add(); })).Render()
```

### FIELDS.CS

```
public ActionResult Index()
{
 return View();
}
```

#### *Applying formatting to measures*

Formatting defines a way in which values should be displayed in pivot table. For example, format “C0” denotes the values should be displayed in currency pattern without decimal points. To do so, define the [PivotViewFormatSetting](#) with its [Name](#) and [Format](#) properties. In this sample, the [Name](#) property is set as “[Measures].[Internet Sales Amount]”, a measure from value axis and its [Format](#) is set as “C0”. Likewise, we can set format for other measures as well.

**Note:** Only measures from [Values](#) axis, which is in the form of numeric data values are applicable for formatting.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("600").DataSourceSettings(dataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product Categories]").Caption("Product Categories").Add(); columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date Fiscal").Add(); })
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("[Measures].[Internet Sales Amount]").Format("C0").Add();
 })
 .Render()
```

### FORMATTING.CS

```
public ActionResult Index()
{
 return View();
}
```

*Enable grouping bar*

The Grouping Bar feature automatically populates [OLAP cube elements](#) from the bound data source and allows end users to drag [OLAP cube elements](#) between different axes such as [Rows](#), [Columns](#), [Values](#) and [Filters](#), and change pivot view at runtime. Sorting, filtering and removing of elements is also possible. It can be enabled by setting the [ShowGroupingBar](#) property to **true** as follows.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").ShowGroupingBar(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })).Render()
<style>
 #pivotview {
 display: block;
 }
</style>
```

**GROUPING-BAR.CS**

```
public ActionResult Index()
{
 return View();
}
```

*Enable pivot field list*

The component provides a built-in Field List similar to Microsoft Excel. It allows you to add or remove [OLAP cube elements](#) and also rearrange the [OLAP cube elements](#) between different axes, including [Rows](#), [Columns](#), [Values](#) and [Filters](#) along with filter and sort options dynamically at runtime. It can be enabled by setting the [ShowFieldList](#) property to **true** as follows.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("400").ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
```

```

 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })).Render()

```

### FIELD-LIST.CS

```

public ActionResult Index()
{
 return View();
}

```

#### Exploring filter axis

The filter axis contains collection of [OLAP cube elements](#) such as hierarchies and calculated members that would act as master filter over the data bound in [Rows](#), [Columns](#) and [Values](#) axes of the pivot table. The [OLAP cube elements](#) along with filter members could be set to filter axis either through report via code behind or by dragging and dropping [OLAP cube elements](#) from other axes to filter axis via grouping bar or field list at runtime.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").ShowFieldList
(true).ShowGroupingBar(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
 .FilterSettings(filterSettings =>
 {
filterSettings.Name("[Date].[Fiscal]").Items(ViewBag.filterMembers).LevelCou
nt(3).Add();
 })).Render()
<style>
 #pivotview {
 display: block;
 }
</style>

```

**EXPLORE-FILTER.CS**

```
public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
 return View();
}
```

*Calculated field*

The calculated field allows user to insert or add a new calculated field based on the available [OLAP cube elements](#) from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.

The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

It can be customized using the [PivotViewCalculatedFieldSetting](#) property through code behind. The setting required for calculate field feature at code behind are:

- [Name](#): It allows to set the unique name for new calculated field.
- [Formula](#): It allows to set the user-defined expression.
- [HierarchyUniqueName](#): It allows to specify dimension unique name whose hierarchies alone should be used in the expression. This will be applicable only for calculated dimension.
- [FormatString](#): It allows to set the format string for the resultant calculated field.

You need to set [IsCalculatedField](#) property to true, while adding calculated fields to respective axis through code behind.

Also calculated fields can be added at run time through the built-in dialog. The dialog can be enabled by setting the [AllowCalculatedField](#) property to **true** as follows. You will see a button enabled in the Field List UI automatically to invoke the calculated field dialog and perform necessary operation.

**Note:** Calculated measure can be added only in value axis.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").ShowFieldList
(true).ShowGroupingBar(true).AllowCalculatedField(true).DataSourceSettings(d
ataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
```

```

Sales Amount]").Caption("Internet Sales Amount").Add(); values.Name("Order
on Discount").IsCalculatedField(true).Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
 .FilterSettings(filterSettings =>
 {

filterSettings.Name("[Date].[Fiscal]").Items(ViewBag.filterMembers).LevelCou
nt(3).Add();
 }).CalculatedFieldSettings(calculatedFieldSettings =>
 {

calculatedFieldSettings.Name("BikeAndComponents").Formula("([Product].[Produ
ct Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components])").HierarchyUniqueName("[Product].[Produ
ct Categories]").FormatString("Standard").Add();
 calculatedFieldSettings.Name("Order on
Discount").Formula("[Measures].[Order Quantity] + ([Measures].[Order
Quantity] * 0.10)").FormatString("Currency").Add();
 })).Render()
<style>
 #pivotview {
 display: block;
 }
</style>

```

### **CALCULATED-FIELD.CS**

```

public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal
Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
 return View();
}

```

Users can add a calculated field at runtime through the built-in dialog by using the following steps.

**Step 1:** Click the "CALCULATED FIELD" button in the field list dialog positioned at the top right corner. The calculated field dialog will be opened now. Enter the name of the calculated field to be created.

<br/>

<br/>

Field List

12 ↑ ↓

All Fields

> Calculated Members

> Measures

> Account

> Customer

> Date

> Delivery Date

> Department

> Destination Currency

Filters

Date Fiscal (All P...

Rows

Customer Geo...

CALCULATED FIELD

Columns

Product Categ...

Measures

Values

Customer Count

Internet Sales Amount

CLOSE

<br/>

<br/>



## Create Calculated Field

All Fields

Calculated Members

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Name

BikeAndComponents

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

**Step 2:** Frame the expression by dragging and dropping the fields from the tree view on the left side of the dialog using simple arithmetic operators. **Example:** "IIF([Measures].[Internet Sales Amount]^0.5 > 100, [Measures].[Internet Sales Amount]\*100, [Measures].[Internet Sales Amount]/100)". Refer here to learn more about the supported [operators](#) and [functions](#) to frame the expression.

&lt;br/&gt;

&lt;br/&gt;

### Create Calculated Field

All Fields

> History

> Large Photo

> Model Name

> Product

> Product Categories

> Product Line

> Product Model Lines

> Style

> Subcategory

> Promotion

> Reseller

> Reseller Sales Order Details

Field Name

BikeAndComponents

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Product Categories

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

**Step 3:** Confirm the type of the field to be created - calculated measure or calculated dimension.

&lt;br/&gt;

&lt;br/&gt;

**Create Calculated Field**

**All Fields**

- Calculated Members
  - BikeAndComponents
  - Order on Discount
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details

**Field Name**  
BikeAndComponents

**Expression**  
([Product].[Product Categories].[Category].[Bikes] + [Product].[Product Categories].[Category].[Components] )

**Field Type**  
Dimension

Measure  
Dimension

**Format String**  
Standard

Enter custom format string

CLEAR OK CANCEL

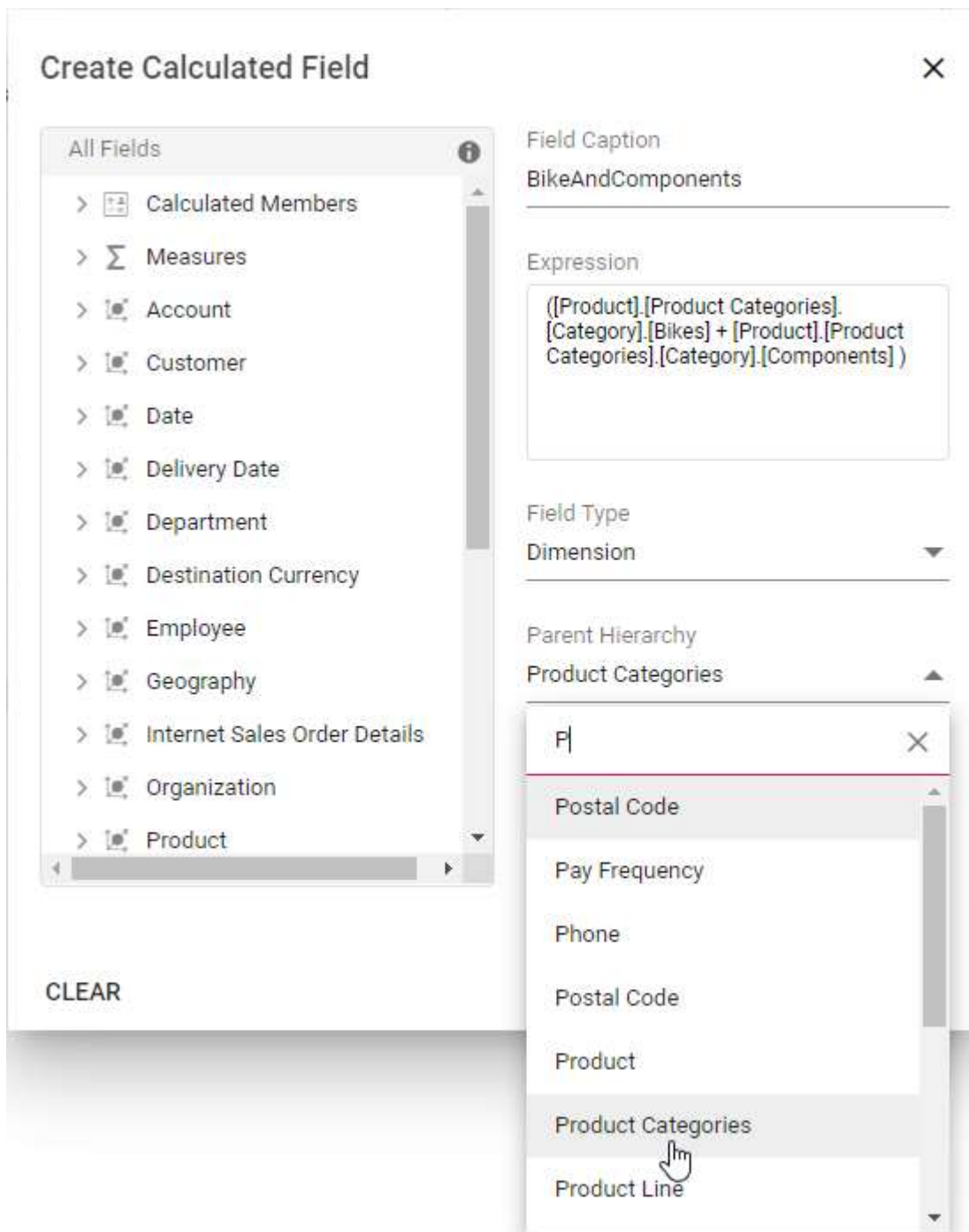
<br/>

<br/>

**Step 4:** Choose the parent hierarchy of the calculated field. NOTE: It is only applicable to the calculated dimension.

<br/>

<br/>



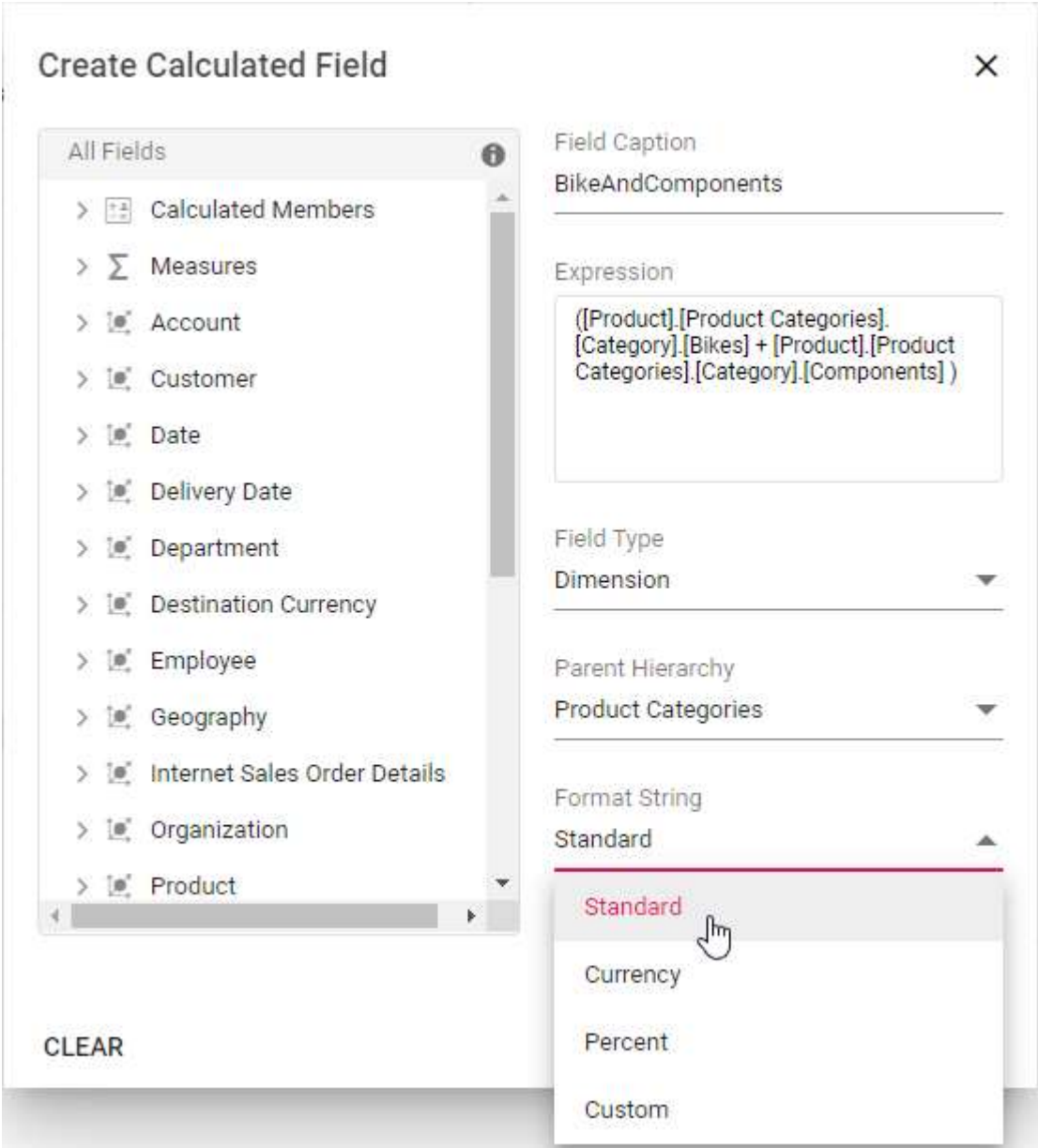
<br/>

<br/>

**Step 5:** Then select the format string from the drop-down list and finally click "OK".

<br/>

<br/>



<br/>

<br/>

|             | Accessories    |                       |                   |
|-------------|----------------|-----------------------|-------------------|
|             | Customer Count | Internet Sales Amount | Order on Discount |
| ▶ Australia | 2,905          | \$138,690.63          | \$68,124.10       |
| ▶ Canada    | 1,230          | \$103,377.85          | \$68,124.10       |
| ▶ France    | 1,505          | \$63,406.78           | \$68,124.10       |
| ▶ Germany   | 1,535          | \$62,232.59           | \$68,124.10       |

<br/>

<br/>

#### Format String

Allows you to specify the required format string while creating new calculated field. Supported format strings are:

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "###0.##0#". This shows the value "9584.3" as "9584.300."

By default, **Standard** will be selected from the drop down list.

## Create Calculated Field

All Fields

Calculated Members

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Product

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Standard

Currency

Percent

Custom

CLEAR

### Renaming the existing calculated field

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing name getting displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

### Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;



## Create Calculated Field

All Fields

Calculated Members

Σ Order on Discount

Σ Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Caption

Order Quantity on Discount

Expression

[Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

### [Editing the existing calculated field formula](#)

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, click the target field. User can now see the existing expression getting displayed in a "Expression" section. Now, change the expression based on user requirement and click "OK".

### Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

**Create Calculated Field**

**All Fields**

- Calculated Members
  - Order on Discount**
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details
- Organization

**Field Caption**  
Order Quantity on Discount

**Expression**  
[Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.50)

**Field Type**  
Measure

**Parent Hierarchy**  
Account Number

**Format String**  
Standard

Enter custom format string

**CLEAR** **OK** **CANCEL**

#### [Reusing the existing formula in a new calculate field](#)

While creating a new calculated field, if user wants to add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Expression" section.

### Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

### Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Caption

Discount Quantity

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Order on Discount

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

**Create Calculated Field**

**All Fields**

- Calculated Members
- Order on Discount**
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details
- Organization

**Field Caption**  
Discount Quantity

**Expression**  
[Measures].[Order on Discount]

**Field Type**  
Measure

**Parent Hierarchy**  
Account Number

**Format String**  
Standard

Enter custom format string

CLEAR OK CANCEL

#### [Modifying the existing format string](#)

Existing calculated field's format string can be modified only through the UI at runtime. To do so, open the calculated field dialog and click the target calculated field. User can now see the format string for the existing calculated field getting displayed in a drop-down list. Change the format string based on the requirement and finally click "OK".

Create Calculated Field

All Fields

Calculated Members

Order on Discount

Measures

Account

Customer

Date

Delivery Date

Department

Destination Currency

Employee

Geography

Internet Sales Order Details

Organization

Field Name

Enter the field name

Expression

Example: [Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

Field Type

Measure

Parent Hierarchy

Account Number

Format String

Standard

Enter custom format string

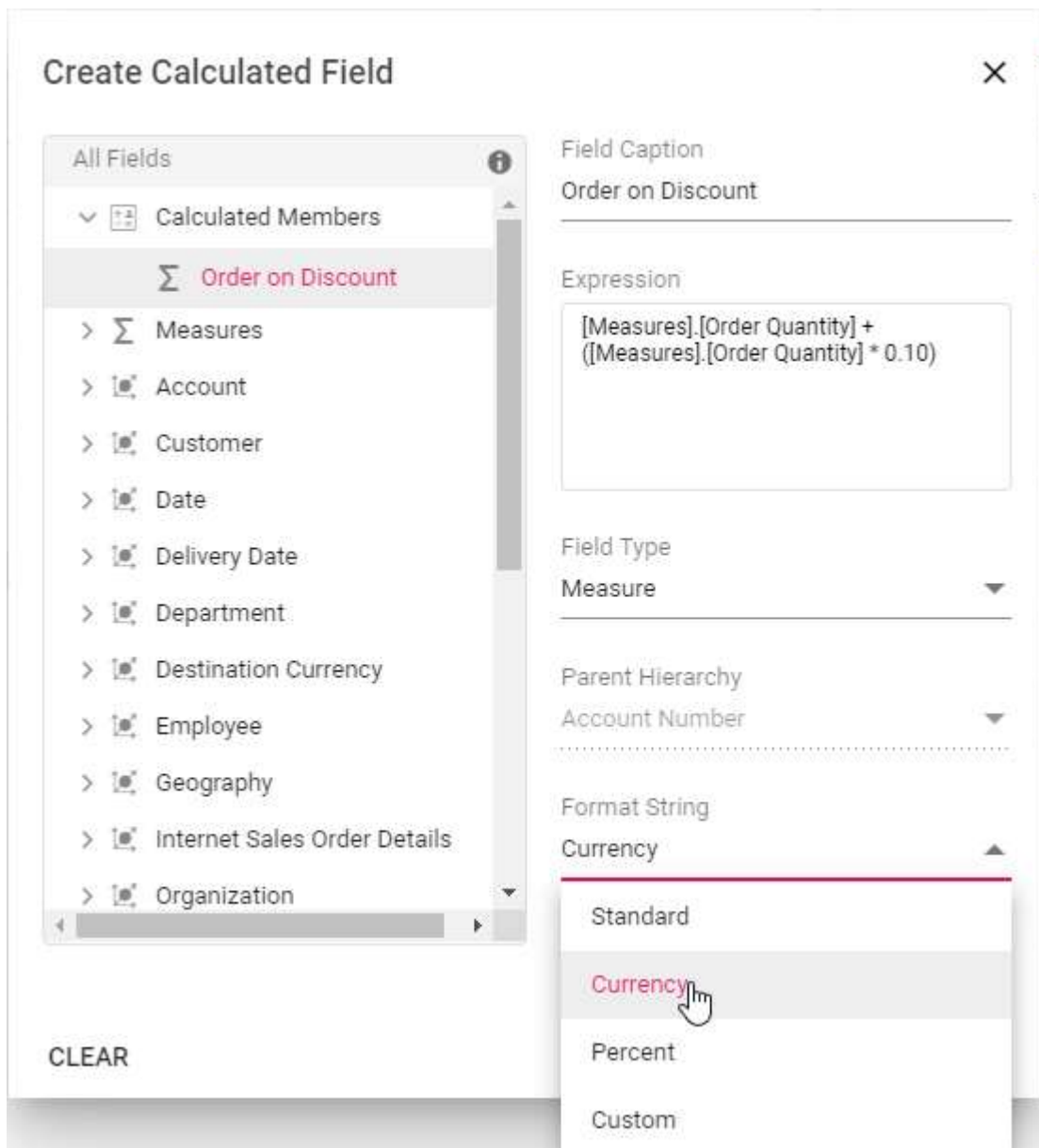
CLEAR

OK

CANCEL

<br/>

<br/>



#### Clearing the changes while editing the calculated field

Previous changes can be cleared by using the "Clear" option while performing operations such as creating and editing the calculated field. To do so, click the "Clear" button in the bottom left corner of the dialog.



**Create Calculated Field**

**All Fields**

- Calculated Members
  - Order on Discount**
  - BikeAndComponents
- Measures
- Account
- Customer
- Date
- Delivery Date
- Department
- Destination Currency
- Employee
- Geography
- Internet Sales Order Details

**Field Caption**  
Order on Discount

**Expression**  
[Measures].[Order Quantity] + ([Measures].[Order Quantity] \* 0.10)

**Field Type**  
Measure

**Parent Hierarchy**  
Account Number

**Format String**  
Currency

Enter custom format string

**CLEAR** **OK** **CANCEL**

### Virtual Scrolling

Allows large amounts of data to be loaded without any performance degradation by rendering rows and columns in relation to the current viewport. Rest of the data will be brought into the viewport dynamically based on vertical or horizontal scroll position. This feature can be enabled by setting the [EnableVirtualization](#) property in [PivotView](#) class to **true**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("600").DataSourceSettings(dataSourceSettings => dataSourceSettings

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
.Rows(rows => {
rows.Name("[Customer].[Customer]").Caption("Customer").Add(); })
```

```

.Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
.Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
.Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
.FormatSettings(formatSettings =>
{
 formatSettings.Name("[Measures].[Internet Sales
Amount]").Format("C0").Add();
})) .EnableVirtualization(true).Render()

```

## VIRTUAL.CS

```

public ActionResult Index()
{
 return View();
}

```

|                   | Bikes          |               |             |               | Clothing      |               |             |               |
|-------------------|----------------|---------------|-------------|---------------|---------------|---------------|-------------|---------------|
|                   | Mountain Bikes |               | Road Bikes  |               | Touring Bikes |               | Clothing    |               |
|                   | Customer C.    | Internet Sal. | Customer C. | Internet Sal. | Customer C.   | Internet Sal. | Customer C. | Internet Sal. |
| James A. Allen    | 1              | \$1,485       |             |               |               |               |             |               |
| James A. Hayes    | 1              | \$2,339       | 1           | \$761         |               |               |             |               |
| James A. Zhang    |                |               | 1           | \$549         |               |               |             |               |
| James Alexander   |                |               |             |               |               |               |             |               |
| James B. Adams    |                |               |             |               |               |               | 1           | \$5           |
| James Bryant      |                |               |             |               |               |               | 1           | \$3           |
| James Sellar      | 1              |               |             |               |               |               |             |               |
| James C. Campbell |                |               | 1           | \$1,120       |               |               |             |               |
| James D. Diaz     |                |               | 1           | \$6,825       |               |               |             |               |
| James E. Scott    |                |               | 1           | \$2,441       |               |               |             |               |
| James C. Yung     |                |               |             |               |               |               | 1           | \$9           |
| James Chen        |                |               |             |               |               |               |             |               |
| James Coleman     |                |               |             |               |               |               |             |               |

### Limitations for virtual scrolling

- The [ColumnWidth](#) property in [GridSettings](#) should be in pixels. The percentage value is not accepted.
- Resizing columns and setting the width of individual columns will affect scrolling and is therefore not recommended.
- The grand totals option is not supported by virtual scrolling.

### Data Binding

To bind OLAP datasource to the pivot table, you need to specify following properties under [DataSourceSettings](#) option.

| Properties | Description |

|-----|-----|

| [Cube](#) | Points the respective cube name from OLAP database. |

| [ProviderType](#) | Points the provider type for pivot table to identify the type of data source. |

| [Url](#) | Contains the cube URL for establishing the connection (online).|

| [Catalog](#) | Contains the database name (catalog name) to fetch the data. |

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("600").DataSourceSettings(dataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product Categories]").Caption("Product Categories").Add(); columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date Fiscal").Add(); })).Render()
```

### DATA-BINDING.CS

```
public ActionResult Index()
{
 return View();
}
```

### *Fields*

#### *Measures in row axis*

By default, the measures are plotted in column axis. You can place measures in row axis either thorough code behind or UI. To plot those measures in row axis, place the **Measures** field in the row axis as follows.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("600").ShowFieldList(true).ShowGroupingBar(true).DataSourceSettings(dataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer Geography]").Caption("Customer Geography").Add(); rows.Name("[Measures]").Caption("Measures").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product Categories]").Caption("Product Categories").Add(); })
 .Values(values => { values.Name("[Measures].[Customer Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date Fiscal").Add(); })
 .FilterSettings(filterSettings =>
```

```

{
filterSettings.Name("[Date].[Fiscal]").Items(ViewBag.filterMembers).LevelCount(3).Add();
))) .Render()
<style>
 #pivotview {
 display: block;
 }
</style>

```

### MEASURES-IN-ROW.CS

```

public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
 return View();
}

```

#### Measures in different position

You can place measures in different position in row or column axis either thorough code behind or UI. In this sample, **Measures** placed before the dimension in the column axis.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("600").DataSourceSettings(dataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product Categories]").Caption("Product Categories").Add(); columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date Fiscal").Add(); })
 .FormatSettings(formatSettings =>
 {
formatSettings.Name("[Date].[Fiscal]").Name("[Measures].[Internet Sales Amount]").Format("C0").Add();
 }
))) .Render()

```

### MEASURES-POSITION.CS

```

public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
}

```

```
return View();
}
```

### Named set

Named set is a multidimensional expression (MDX) that returns a set of dimension members, which can be created by combining the cube data, arithmetic operators, numbers, and functions.

You can bind the named sets in the pivot table by setting its unique name in the [Name](#) property either in row or column axis and [IsNamedSet](#) boolean property to **true**. In this sample, we have added "Core Product Group" named set in the column axis.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").ShowFieldList
(true).ShowGroupingBar(true).AllowCalculatedField(true).DataSourceSettings(d
ataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
.Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
.Columns(columns => { columns.Name("[Core Product
Group]").Caption("Core Product Group").IsNamedSet(true).Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
.Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
.Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
.FilterSettings(filterSettings =>
{

filterSettings.Name("[Date].[Fiscal]").Items(ViewBag.filterMembers).LevelCou
nt(3).Add();
})) .Render()
<style>
 #pivotview {
 display: block;
 }
</style>
```

### NAMED-SET.CS

```
public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal
Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
 return View();
}
```

### Configuring authentication

Users can configure basic authentication information to access the OLAP cube using the [Authentication](#) property. The settings required to configure are as follows:

- **UserName:** It allows the user to set a username that recognizes the basic authentication of the IIS.
- **Password:** It allows to set the appropriate password.

**Note:** If the user does not configure the authentication, a default popup will appear in the browser to get the authentication information.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").DataSourceSettings(dataSourceSettings => dataSourceSettings.EnableSorting(true)
 .Authentication(new PivotViewAuthentication {
 UserName = "UserName",
 Password = "Password" })

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })).Render()
```

### AUTHENTICATION.CS

```
public ActionResult Index()
{
 return View();
}
```

### *Roles*

SQL Server Analysis Services uses [Roles](#) to limit data access within a cube. Each role defines a set of permissions that can be granted to a single user or groups of users. It is used to manage security by limiting access to sensitive data and determining who has access to and can change the cube. It can be configured using the [Roles](#) property in [DataSourceSettings](#).

The [Roles](#) property can be used to specify one or more roles to the OLAP cube, separated by commas.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").DataSourceSettings(dataSourceSettings => dataSourceSettings
 .Roles("Role1")

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
```

```

 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); })
 .Render()

```

## ROLES.CS

```

public ActionResult Index()
{
 return View();
}

```

## OLAP Cube: Elements

### Field list

The field list, aka cube dimension browser, is a tree view like structure that organizes the cube elements such as dimensions, hierarchies, measures, etc., from the selected cube into independent logical groups.

### Types of node in field list

- **Display folder:** A folder that contains a set of similar elements.
- **Measure:** Quantity available for analysis.
- **Dimension:** A name given to the parts of the cube that categorizes data.
- **Attribute Hierarchy:** Level of attributes down the hierarchy.
- **User-defined Hierarchy:** Members of a dimension in a hierarchical structure.
- **Level:** Denotes a specific level in the category.
- **Named Set:** A collection of tuples and members, that can be defined and saved as a part of cube definition for later use.

### Measure

In a cube, a measure is a set of values that are based on a column in the cube's fact table and are usually numeric. The measures are the central values of a cube that are analyzed. That is, measures are the numeric data of primary interest to users browsing a cube. You can select measures depend on the types of users request. Some common measures are sales, costs, expenditures, and production count.

### Dimension

A simple dimension object is composed of basic information such as name, hierarchy, level, and members. You can create a dimension element by specifying its name and providing the hierarchy and level name. The dimension element contains the hierarchical details and information about each included level elements in that hierarchy. A hierarchy can have any number of level elements and the level elements can have any number of members and the member elements can have any number of child members.

### Hierarchy

Each element of a dimension can be summarized using a hierarchy. The hierarchy is a series of parent-child relationship, where a parent member represents the consolidation of members which are its children. Parent members can be further aggregated as the children of another parent. For example,

May 2005 can be summarized into Second Quarter 2005 which in turn would be summarized in the year 2005.

#### Level

Level element is the child of hierarchy element which contains a set of members, each of which has the same rank within a hierarchy.

#### Attribute hierarchy

Attribute hierarchy contains the following levels:

- A leaf level contains distinct attribute member, and each member of the leaf level is known as a leaf member.
- Intermediate levels if the attribute hierarchy is a parent-child hierarchy.
- An optional (all) level contains the aggregated value of the attribute hierarchy's leaf members, with the member of the (all) level also known as the (all) member.

#### User-defined hierarchy

User-defined hierarchy organizes the members of a dimension into hierarchical structure and provides navigation paths in a cube. For example, take a dimension table that supports three attributes such as year, quarter, and month. The year, quarter, and month attributes are used to construct a user-defined hierarchy, named Calendar, in the time dimension that relates to all levels.

#### Differentiating user-defined hierarchy and attribute hierarchy

- User-defined hierarchy contains more than one level whereas attribute hierarchy contains only one level.
- User-defined hierarchy provides the navigation path between the levels taken from attribute hierarchies of the same dimension.
- The attribute hierarchy and the user-defined hierarchy are represented in different ways as shown in the following table.

#### Named set

A named set is a collection of tuples and members, which can be defined and saved as a part of the cube definition. Named set records reside inside the sets folder, which is under a dimension element. These elements can be dragged to [Rows](#) or [Columns](#) axis via grouping bar or field list at runtime. To work with a lengthy, complex, or commonly used expression easier, Multidimensional Expressions (MDX) allows you to define a named set.

#### Calculated field

The calculated field allows user to insert or add a new calculated field based on the available OLAP cube elements from the bound data source. Calculated fields are nothing but customized dimensions or measures that are newly created based on the user-defined expression.










The two types of calculated fields are as follows:

- **Calculated Measure** – Creates a new measure through user-defined expression.
- **Calculated Dimension** – Creates a new dimension through user-defined expression.

#### Symbolic representation of the nodes inside field list

| Icon | Name | Node type | Is Draggable |



|  |                                                                                   |                                                                                       |                                                                                       |                                        |
|--|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|----------------------------------------|
|  | -----                                                                             | -----                                                                                 | -----                                                                                 | -----                                  |
|  |  | Display Folder                                                                        | Display Folder                                                                        | False                                  |
|  |  | Measure                                                                               | Measure                                                                               | False                                  |
|  |  | Dimension                                                                             | Dimension                                                                             | False                                  |
|  |  | User Defined Hierarchy                                                                | Hierarchy                                                                             | True                                   |
|  |  | Attribute Hierarchy                                                                   | Hierarchy                                                                             | True                                   |
|  |  | <br> | <br> | Levels (in order)  Level Element  True |
|  |  | Named Set                                                                             | Named Set                                                                             | True                                   |

**Note:** In general, the Pivot Table is created using the built-in engine for given data source. This is an optional feature that allows you to create the Pivot Table with a server-side pivot engine and external data binding. And this option is applicable only for relational data source.

### Getting Started with Syncfusion Server-side Pivot Engine

This section briefs the Syncfusion assembly [Syncfusion.Pivot.Engine](#), which is used in a server-side application to perform all Pivot calculations such as aggregation, filtering, sorting, grouping, and so on, and only the information to be displayed in the Pivot Table's viewport is passed to the client-side (browser) via web service (Web API) rather than the entire data source. It reduces network traffic and improves the rendering performance of the Pivot Table, especially when dealing with large amounts of data. It also works best with virtual scrolling enabled and supports all the Pivot Table's existing features.

#### Quick steps to render the Pivot Table by using the server-side Pivot Engine

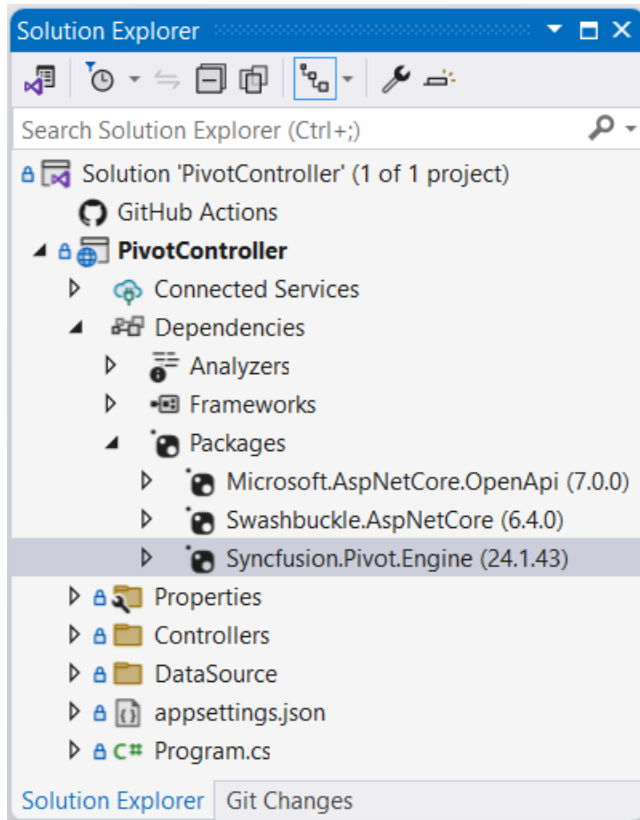
##### *Download and installing Server-side Pivot Engine*

1. Download the ASP.NET Core-based stand-alone Pivot Table [application](#) from the GitHub repository.

2. The **PivotController** (Server-side) application that is downloaded includes the following files.

- **PivotController.cs** file under **Controllers** folder – This helps to do data communication with Pivot Table.
- **DataSource.cs** file under **DataSource** folder – This file has model classes to define the structure of the data sources.
- The sample data source files **sales.csv** and **sales-analysis.json** under **DataSource** folder.

3. Open the **PivotController** application in Visual Studio where the Syncfusion library [Syncfusion.Pivot.Engine](#) will be downloaded automatically from the nuget.org site.



#### Connecting Pivot Table to Server-side Pivot Engine

1. Run the **PivotController** (Server-side) application which will be hosted in IIS shortly.
2. Then in the Pivot Table sample, set the [mode](#) property under [e-datasourcesettings](#) as **Server** and map the URL of the hosted Server-side application in [URL](#) property of `dataSourceSettings`.

```
`html
```

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
```

```
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
```

```
).Render()
```

```
`
```

3. Frame and set the report based on the data source available in the **PivotController** application.

```
`html
```

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
```

```
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
```

```
.FormatSettings(formatsettings =>
```

```
{
```

```
formatsettings.Name("Price").Format("C").Add();
```

```

}).Rows(rows =>
{
rows.Name("ProductID").Add();
}).Columns(columns =>
{
columns.Name("Year").Add();
}).Values(values =>
{
values.Name("Sold").Add();
values.Name("Amount").Add();
})
).Render()
`

```

4. Run the sample to get the following result.

|        | FY 2015    |                | FY 2016    |                | FY 2017    |
|--------|------------|----------------|------------|----------------|------------|
|        | Units Sold | Sold Amount    | Units Sold | Sold Amount    | Units Sold |
| PRO-10 | 7901220    | \$1,841,712.00 | 7561880    | \$1,762,804.00 | 7591220    |
| PRO-11 | 7174625    | \$1,673,533.00 | 7338175    | \$1,711,599.80 | 7164625    |
| PRO-12 | 9168140    | \$2,135,848.80 | 7689400    | \$1,792,704.00 | 7375140    |
| PRO-13 | 6674280    | \$1,560,619.20 | 8585260    | \$2,001,738.40 | 6574280    |
| PRO-14 | 7489240    | \$1,745,841.20 | 7167840    | \$1,675,479.20 | 7390240    |
| PRO-15 | 8149185    | \$1,902,397.00 | 7450585    | \$1,734,093.00 | 7249185    |
| PRO-16 | 7070150    | \$1,652,838.40 | 8037500    | \$1,873,597.60 | 7880150    |

## Available configurations in Server-side application

### Supportive Data Sources

The server-side Pivot Engine supports the following data sources,

- Collection
- JSON
- CSV
- DataTable
- Dynamic

### Collection

The collection data sources such as List, IEnumerable, and so on are supported. This can be bound using the **GetData** controller method. Also, in the Pivot Table sample, set the [type](#) property under [e-datasourcesettings](#) to **JSON**, which is also the default enumeration value.

In the server-side application (**PivotController**), a collection type data source is framed in the **DataSource.cs** file as shown in the following.

```
`csharp
public class PivotViewData
{
 public string ProductID { get; set; }
 public string Country { get; set; }
 public string Product { get; set; }
 public double Sold { get; set; }
 public double Price { get; set; }
 public string Year { get; set; }
 public List<PivotViewData> GetVirtualData()
 {
 List<PivotViewData> VirtualData = new List<PivotViewData>();
 for (int i = 1; i <= 10000; i++)
 {
 PivotViewData p = new PivotViewData
 {
 ProductID = "PRO-" + ((100 + i)%20),
 Year = (new string[] { "FY 2015", "FY 2016", "FY 2017", "FY 2018", "FY 2019" })[new Random().Next(5)],
 Country = (new string[] { "Canada", "France", "Australia", "Germany", "France" })[new
 Random().Next(5)],
 Product = (new string[] { "Car", "Van", "Bike", "Flight", "Bus" })[new Random().Next(5)],
 Price = (3.4 * i) + 500,
 Sold = (i * 15) + 10
 };
 VirtualData.Add(p);
 }
 return VirtualData;
 }
}
```

To bind the data source, set its model type **PivotViewData** to **TValue** of the **PivotEngine** class.

```
`csharp
```

```
private PivotEngine<DataSource.PivotViewData> PivotEngine = new
PivotEngine<DataSource.PivotViewData>();
`
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
`csharp
public async Task<object> GetData(FetchData param)
{
 return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
 async (cacheEntry) =>
 {
 cacheEntry.SetSize(1);
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 // Here bind the collection type data source.
 return new DataSource.PivotViewData().GetVirtualData();
 });
}
`
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```
`html
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Price").Format("C").Add();
}).Rows(rows =>
{
 rows.Name("ProductID").Add();
}).Columns(columns =>
{
 columns.Name("Year").Add();
}).Values(values =>
{
 values.Name("Sold").Add();
}
```

```

values.Name("Amount").Add();
})
).Render()
`

```

|        | FY 2015    |                | FY 2016    |                | FY 2017    |
|--------|------------|----------------|------------|----------------|------------|
|        | Units Sold | Sold Amount    | Units Sold | Sold Amount    | Units Sold |
| PRO-10 | 7901220    | \$1,841,712.00 | 7561880    | \$1,762,804.00 | 7591220    |
| PRO-11 | 7174625    | \$1,673,533.00 | 7338175    | \$1,711,599.80 | 7164625    |
| PRO-12 | 9168140    | \$2,135,848.80 | 7689400    | \$1,792,704.00 | 7378140    |
| PRO-13 | 6674280    | \$1,560,619.20 | 8585260    | \$2,001,738.40 | 6574280    |
| PRO-14 | 7489240    | \$1,745,841.20 | 7167840    | \$1,675,479.20 | 7392400    |
| PRO-15 | 8149185    | \$1,902,397.00 | 7450585    | \$1,734,093.00 | 7249185    |
| PRO-16 | 7070150    | \$1,652,838.40 | 8037500    | \$1,873,597.60 | 7880150    |

### JSON

The JSON data from a local \*.json file type can be connected to the Pivot Table. Here, the file can be read by the **StreamReader** option, which will give the result in the string format. The resultant string needs to be converted to collect data that can be bound to the Server-side pivot engine.

In the Server-side application, **sales-analysis.json** file is available under **DataSource** folder and its model type is defined in **DataSource.cs** file.

```

`csharp
public class PivotJSONData
{
 public string Date { get; set; }
 public string Sector { get; set; }
 public string EnerType { get; set; }
 public string EneSource { get; set; }
 public int PowUnits { get; set; }
 public int ProCost { get; set; }
 public List<PivotJSONData> ReadJSONData(string url)
 {
 WebClient myWebClient = new WebClient();
 Stream myStream = myWebClient.OpenRead(url);
 StreamReader stream = new StreamReader(myStream);
 string result = stream.ReadToEnd();
 stream.Close();
 return Newtonsoft.Json.JsonConvert.DeserializeObject<List<PivotJSONData>>(result);
 }
}

```

```

}
}
`

```

To bind the data source, set its model type **PivotJSONData** to **TValue** of the **PivotEngine** class.

```

`csharp
private PivotEngine<DataSource.PivotJSONData> PivotEngine = new PivotEngine<DataSource.
PivotJSONData>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`csharp
public async Task<object> GetData(FetchData param)
{
 return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
 async (cacheEntry) =>
 {
 cacheEntry.SetSize(1);
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 // Here bind JSON type data source from the sales-analysis.json file.
 return new DataSource.PivotJSONData().ReadJSONData(_hostingEnvironment.ContentRootPath +
 "\\DataSource\\sales-analysis.json");
 });
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`html
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
.Type(Syncfusion.EJ2.PivotView.DataSourceType.CSV)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("ProCost").Format("C").Add();
}).Rows(rows =>
{

```

```

rows.Name("EneSource").Add();
}).Columns(columns =>
{
columns.Name("EnerType").Add();
}).Values(values =>
{
values.Name("PowUnits").Add();
values.Name("ProCost").Add();
})
).Render()
,

```

|                | Biomass    |             | Free Energy |             | Grand Total |
|----------------|------------|-------------|-------------|-------------|-------------|
|                | Units Sold | Sold Amount | Units Sold  | Sold Amount | Units Sold  |
| Bio-diesel     | 1042       | \$1,439.00  |             |             | 1042        |
| Ethanol Fuel   | 595        | \$1,031.00  |             |             | 595         |
| Geo-thermal    |            |             | 1528        | \$2,115.00  | 1528        |
| Hydro-electric |            |             | 3378        | \$3,244.00  | 3378        |
| Solar          |            |             | 7929        | \$6,210.00  | 7929        |
| Wastage        | 712        | \$1,043.00  |             |             | 712         |
| Wind           |            |             | 15571       | \$7,666.00  | 15571       |

JSON data from any remote server, like a local JSON file, can also be supported. It accepts both directly downloadable files (*.json*) and *web service URLs*. To bind this, the URL of the *.json* file of a remote server has to be mapped under the **GetData** method. The rest of the configurations are the same as described above.

In the server-side application, the CDN link is used to connect the same **sales-analysis.json** file which is already hosted in the Syncfusion server.

```

`csharp
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind JSON type data source from remote server.

```



```
return new DataSource.PivotJSONData().ReadJSONData("http://cdn.syncfusion.com/data/sales-
analysis.json");
});
}
`
```

### CSV

The CSV data from a local \*.csv file type can be connected to the Pivot Table. Here, the file can be read by the **StreamReader** option, which will give the result in the string format. The resultant string needs to be converted to collect data that can be bound to the server-side pivot engine. Also, in the Pivot Table sample, set the [type](#) property under [e-datasourcesettings](#) as **CSV**.

In the server application, the **sales.csv** file is available under the **DataSource** folder, and its model type is defined in the **DataSource.cs** file.

```
`csharp
public class PivotCSVData
{
 public string Region { get; set; }
 public string Country { get; set; }
 public string ItemType { get; set; }
 public string SalesChannel { get; set; }
 public string OrderPriority { get; set; }
 public string OrderDate { get; set; }
 public int OrderID { get; set; }
 public string ShipDate { get; set; }
 public int UnitsSold { get; set; }
 public double UnitPrice { get; set; }
 public double UnitCost { get; set; }
 public double TotalRevenue { get; set; }
 public double TotalCost { get; set; }
 public double TotalProfit { get; set; }
 public List<string[]> ReadCSVDData(string url)
 {
 List<string[]> data = new List<string[]>();
 using (StreamReader reader = new StreamReader(new WebClient().OpenRead(url)))
 {
 string line;
```

```

while ((line = reader.ReadLine()) != null)
{
 line = line.Trim();
 if (!string.IsNullOrEmpty(line))
 {
 data.Add(line.Split(','));
 }
}
return data;
}
}
}
`

```

To bind the data source, set its model type **PivotCSVData** to **TValue** of the **PivotEngine** class.

```

`csharp
private PivotEngine<DataSource.PivotCSVData> PivotEngine = new PivotEngine<DataSource.
PivotCSVData>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`csharp
public async Task<object> GetData(FetchData param)
{
 return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
 async (cacheEntry) =>
 {
 cacheEntry.SetSize(1);
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 // Here bind CSV type data source from sales.csv file.
 return new DataSource.PivotCSVData().ReadCSVData(_hostingEnvironment.ContentRootPath +
 "\\DataSource\\sales.csv");
 });
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`html
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
.Type(Syncfusion.EJ2.PivotView.DataSourceType.CSV)
.FormatSettings(formatsettings =>
{
formatsettings.Name("UnitPrice").Format("C").Add();
}).Rows(rows =>
{
rows.Name("ItemType").Add();
}).Columns(columns =>
{
columns.Name("Region").Add();
}).Values(values =>
{
values.Name("UnitsSold").Add();
values.Name("UnitPrice").Add();
})
).Render()
,

```

|           | Asia       |             | Australia and Oceania |             | Central Ar |
|-----------|------------|-------------|-----------------------|-------------|------------|
|           | Units Sold | Sold Amount | Units Sold            | Sold Amount | Units Sold |
| Baby Food | 55810      | \$3,318.64  | 21548                 | \$765.84    |            |
| Beverages | 84400      | \$806.65    | 34486                 | \$332.15    |            |
| Cereal    | 32668      | \$1,439.90  | 43195                 | \$1,645.60  |            |
| Clothes   | 24290      | \$764.96    | 17020                 | \$655.68    |            |
| Cosmetics | 84630      | \$7,432.40  | 35220                 | \$3,060.40  |            |
| Fruits    | 35679      | \$74.64     | 29951                 | \$46.65     |            |
| Household | 28166      | \$4,009.62  | 55444                 | \$6,014.43  |            |

CSV data from any remote server, like a local CSV file, can also be supported. It accepts both directly downloadable files (.csv) and web service URLs. To bind this, the URL of the .csv file of a remote server has to be mapped under **GetData** method. The rest of the configurations are the same as described above.

In the server application, the CDN link is used to connect the same **sales.csv** file which is already hosted in the Syncfusion server.

```
`csharp
public async Task<object> GetData(FetchData param)
{
 return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
 async (cacheEntry) =>
 {
 cacheEntry.SetSize(1);
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 // Here bind CSV type data source from remote server.
 return new DataSource.PivotCSVData().ReadCSVData("http://cdn.syncfusion.com/data/sales-
analysis.csv");
 });
}
```

#### [DataTable](#)

In the server-side application, there is a manually created DataTable **BusinessObjectsDataView** by mapping the model type **PivotViewData** in **DataSource.cs** file.

```
`csharp
public class BusinessObjectsDataView
{
 public DataTable GetDataTable()
 {
 DataTable dt = new DataTable("BusinessObjectsDataTable");
 PropertyDescriptorCollection pdc = TypeDescriptor.GetProperties(typeof(PivotViewData));
 foreach (PropertyDescriptor pd in pdc)
 {
 dt.Columns.Add(new DataColumn(pd.Name, pd.PropertyType));
 }
 List<PivotViewData> list = new PivotViewData().GetVirtualData();
 foreach (PivotViewData bo in list)
 {
 DataRow dr = dt.NewRow();
 foreach (PropertyDescriptor pd in pdc)
 {
```

```

dr[pd.Name] = pd.GetValue(bo);
}
dt.Rows.Add(dr);
}
return dt;
}
}
`

```

To bind the data source, set its model type **PivotViewData** to **TValue** of the **PivotEngine** class.

```

`csharp
private PivotEngine<DataSource.PivotViewData> PivotEngine = new
PivotEngine<DataSource.PivotViewData>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`csharp
public async Task<object> GetData(FetchData param)
{
return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
async (cacheEntry) =>
{
cacheEntry.SetSize(1);
cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
// Here bind the DataTable.
return new DataSource.BusinessObjectsDataView().GetDataTable();
});
}
`

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```

`html
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
.FormatSettings(formatsettings =>
{

```

```

formatsettings.Name("Price").Format("C").Add();
}).Rows(rows =>
{
rows.Name("ProductID").Add();
}).Columns(columns =>
{
columns.Name("Year").Add();
}).Values(values =>
{
values.Name("Sold").Add();
values.Name("Price").Add();
})
).Render()
,

```

|        | FY 2015    |                | FY 2016    |                | FY 2017    |
|--------|------------|----------------|------------|----------------|------------|
|        | Units Sold | Sold Amount    | Units Sold | Sold Amount    | Units Sold |
| PRO-10 | 7901220    | \$1,841,712.00 | 7561880    | \$1,762,804.00 | 7591       |
| PRO-11 | 7174625    | \$1,673,533.00 | 7338175    | \$1,711,599.80 | 7164       |
| PRO-12 | 9168140    | \$2,135,848.80 | 7689400    | \$1,792,704.00 | 7375       |
| PRO-13 | 6674280    | \$1,560,619.20 | 8585260    | \$2,001,738.40 | 6571       |
| PRO-14 | 7489240    | \$1,745,841.20 | 7167840    | \$1,675,479.20 | 7396       |
| PRO-15 | 8149185    | \$1,902,397.00 | 7450585    | \$1,734,093.00 | 7247       |
| PRO-16 | 7070150    | \$1,652,838.40 | 8037500    | \$1,873,597.60 | 7885       |

### Dynamic

The model type has to be defined in the aforementioned data sources. However, there is no need to define a model type for the following data sources, which are also supported by the server-side pivot engine.

### ExpandoObject

In the server-side application, an **ExpandoObject** type data source is available under the class **PivotExpandoData** in **DataSource.cs** file.

```

`csharp
public class PivotExpandoData
{
public List<ExpandoObject> Orders { get; set; } = new List<ExpandoObject>();
public List<ExpandoObject> GetExpandoData()
{

```

```

Orders = Enumerable.Range(1, 75).Select((x) =>
{
 dynamic d = new ExpandoObject();
 d.OrderID = 1000 + (x % 100);
 d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID" })[new
Random().Next(5)];
 d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
 d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new DateTime(2018, 10, 3), new
DateTime(1995, 9, 9), new DateTime(2012, 8, 2), new DateTime(2015, 4, 11) })[new Random().Next(5)];
 d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
 d.Verified = (new bool[] { true, false })[new Random().Next(2)];
 return d;
}).Cast<ExpandoObject>().ToList<ExpandoObject>();
return Orders;
}
}
`

```

To bind the data source, set its model type as **ExpandoObject** to **TValue** of the **PivotEngine** class.

```

`csharp
private PivotEngine<ExpandoObject> PivotEngine = new PivotEngine<ExpandoObject>();
`

```

Then call the data source in **GetData** method of **PivotController.cs** file.

```

`csharp
public async Task<object> GetData(FetchData param)
{
 return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
 async (cacheEntry) =>
 {
 cacheEntry.SetSize(1);
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 // Here returns ExpandoObject type data source.
 return new DataSource.PivotExpandoData().GetExpandoData();
 });
}

```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```
`html
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
.Rows(rows =>
{
rows.Name("CustomerID").Add();
}).Columns(columns =>
{
columns.Name("ShipCountry").Add();
}).Values(values =>
{
values.Name("Freight").Add();
})
).Render()
`,`
```

|             | UK         | USA        | Grand Total |
|-------------|------------|------------|-------------|
|             | Units Sold | Units Sold | Units Sold  |
| ALFKI       | 1187       | 738        | 1925        |
| ANANTR      | 1011       | 814        | 1825        |
| ANTON       | 1046       | 632        | 1678        |
| BLONP       | 1080       | 749        | 1829        |
| BOLID       | 492        | 929        | 1421        |
| Grand Total | 4816       | 3862       | 8678        |

### Dynamic Objects

In the server-side application, a data source is framed by dynamic objects which is available under the class **PivotDynamicData** in the **DataSource.cs** file.

```
`csharp
public class PivotDynamicData
{
public List<DynamicDictionary> Orders = new List<DynamicDictionary>() { };
public List<DynamicDictionary> GetDynamicData()
{
Orders = Enumerable.Range(1, 100).Select((x) =>
```



```

{
dynamic d = new DynamicDictionary();
d.OrderID = 100 + x;
d.CustomerID = (new string[] { "ALFKI", "ANANTR", "ANTON", "BLONP", "BOLID" })[new
Random().Next(5)];
d.Freight = (new double[] { 2, 1, 4, 5, 3 })[new Random().Next(5)] * x;
d.OrderDate = (new DateTime[] { new DateTime(2010, 11, 5), new DateTime(2018, 10, 3), new
DateTime(1995, 9, 9), new DateTime(2012, 8, 2), new DateTime(2015, 4, 11) })[new Random().Next(5)];
d.ShipCountry = (new string[] { "USA", "UK" })[new Random().Next(2)];
d.Verified = (new bool[] { true, false })[new Random().Next(2)];
return d;
}).Cast<DynamicDictionary>().ToList<DynamicDictionary>();
return Orders;
}

public class DynamicDictionary : System.Dynamic.DynamicObject
{
Dictionary<string, object> dictionary = new Dictionary<string, object>();
public override bool TryGetMember(GetMemberBinder binder, out object result)
{
string name = binder.Name;
return dictionary.TryGetValue(name, out result);
}
public override bool TrySetMember(SetMemberBinder binder, object value)
{
dictionary[binder.Name] = value;
return true;
}
//The "GetDynamicMemberNames" method of the "DynamicDictionary" class must be overridden and
return the property names to perform data operation and editing while using dynamic objects.
public override System.Collections.Generic.IEnumerable<string> GetDynamicMemberNames()
{
return this.dictionary?.Keys;
}
}

```

```
}
,
```

To bind the data source, set its class **PivotDynamicData** to **TValue** of the **PivotEngine** class.

```
`csharp
private PivotEngine<DataSource.PivotDynamicData> PivotEngine = new
PivotEngine<DataSource.PivotDynamicData>();
,
```

Then call the data source in **GetData** method of **PivotController.cs** file.

```
`csharp
public async Task<object> GetData(FetchData param)
{
 return await _cache.GetOrCreateAsync("dataSource" + param.Hash,
 async (cacheEntry) =>
 {
 cacheEntry.SetSize(1);
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 // Here bind data source with dynamic objects.
 return new DataSource.PivotDynamicData().GetDynamicData();
 });
}
,
```

Finally set the appropriate report to the Pivot Table sample based on the above data source.

```
`html
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.Url("https://localhost:44350/api/pivot/post")
.Mode(Syncfusion.EJ2.PivotView.RenderMode.Server)
.Rows(rows =>
{
 rows.Name("CustomerID").Add();
}).Columns(columns =>
{
 columns.Name("ShipCountry").Add();
}).Values(values =>
{
```

```

values.Name("Freight").Add();
})
).Render()
,

```

|             | UK         | USA        | Grand Total |
|-------------|------------|------------|-------------|
|             | Units Sold | Units Sold | Units Sold  |
| ALFKI       | 1030       | 742        | 1772        |
| ANANTR      | 1352       | 1018       | 2370        |
| ANTON       | 2525       | 1782       | 4307        |
| BLONP       | 1786       | 1002       | 2788        |
| BOLID       | 1409       | 2012       | 3421        |
| Grand Total | 8102       | 6556       | 14658       |

### Controller Configuration

#### Memory Cache

In the server-side application, the [Memory Cache](#) option is used to store the data source and engine properties in RAM, which will be used for UI operations. To improve performance, this limits the execution of all initial rendering code to regenerate the aggregated values during each UI operation. The codes below show how we use the memory cache option in the **GetEngine** method to store engine properties.

```

`csharp
public async Task<EngineProperties> GetEngine(FetchData param)
{
 isRendered = false;
 // Engine properties are stored in memory cache with GUID "param.Hash".
 return await _cache.GetOrCreateAsync("engine" + param.Hash,
 async (cacheEntry) =>
 {
 isRendered = true;
 cacheEntry.SetSize(1);
 // Memory cache expiration time can be set here.
 cacheEntry.AbsoluteExpiration = DateTimeOffset.UtcNow.AddMinutes(60);
 PivotEngine.Data = await GetData(param);
 return await PivotEngine.GetEngine(param);
 });
}
,

```

The engine properties are stored in RAM as a cache with a unique ID (GUID) that is transferred from the client-side source code. The GUID is generated at random and will be changed if the page containing the Pivot Table is refreshed or opened in a new tab/window. As a result, each GUID's memory cache contains unique information, and the component operates independently.

Meanwhile, the memory cache is set to expire after 60 minutes from RAM to free its memory. If the component is still running, the data will be generated and stored for another 60 minutes.

#### Methods and its needs

- **Post:** Allows to get the information from the client-side source and calls appropriate controller methods.
- **GetEngine:** Allows to store the engine properties in RAM as a cache which fires on initial rendering or when the memory cache is expired.
- **GetData:** Allows to store data source in RAM as a cache which fires on initial rendering or when the memory cache is expired.
- **GetMembers:** Allows to get the members of a field. This fires when the member editor is opened to do a filtering operation.
- **GetRawData:** Allows to get raw data of an aggregated value cell. This fires when the drill-through or editing dialog is opened.
- **GetPivotValues:** Allows to update the stored engine properties in-memory cache and returns the aggregated values to browser to render the Pivot Table. Here, the return value can be modified. The Pivot Table will be rendered browser-based on this.

#### Pivot Chart in ASP.NET MVC Pivot Table Component

In pivot table component, pivot chart would act as an additional visualization component with its basic and important characteristic like drill down and drill up, 15+ chart types, series customization, axis customization, legend customization, export, print and tooltip. Its main purpose is to show the pivot data in graphical format.

If user prefers, the pivot chart component can also be displayed individually with pivot values and can change the report dynamically with the help of field list and grouping bar. Using the [PivotViewDisplayOption](#) property in [PivotView](#) class, user can set the visibility of grid and chart in pivot table component. It holds below properties,

- [View](#): Specifies the pivot table component to display grid alone or chart alone or both.
- [Primary](#): Specifies the pivot table to display either grid or chart as primary component during initial loading. It is applicable only when setting the property [View](#) to [View.Both](#).

The below sample displays the pivot chart component based on the pivot report bound on it.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
```

```

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()

```

### DISPLAYVIEW.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Data Binding

End user can bind both local and remote data binding options available in the component to feed the data. The [DataSource](#) property can be assigned either with an instance of `DataManager` or list of object.

For more information [refer](#) here.

### Chart Types

Supports 21 different types of charts as follows,

- Line
- Column
- Area
- Bar
- StepArea
- StackingLine
- StackingColumn
- StackingArea
- StackingBar
- StepLine
- Pareto
- Bubble
- Scatter
- Spline
- SplineArea

- StackingLine100
- StackingColumn100
- StackingBar100
- StackingArea100
- Polar
- Radar

[ChartSeriesType.Line](#) is the default pivot chart type. User can change the pivot chart type by using the property [Type](#) in [PivotChartSeries](#) class.

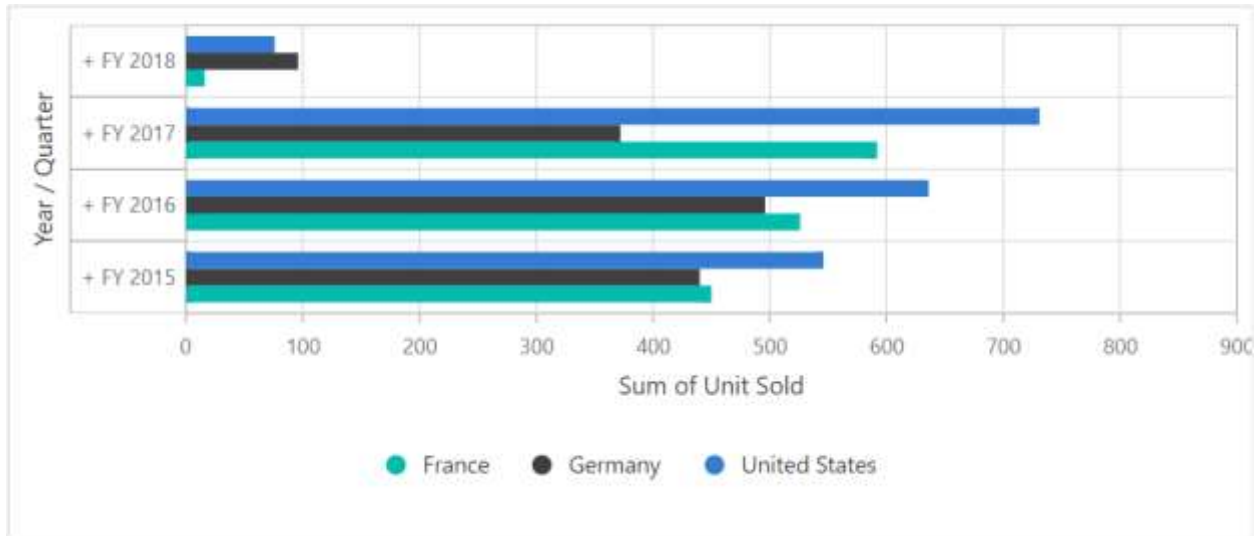
In the below code sample, the pivot chart type is set as [ChartSeriesType.Bar](#).

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Bar))).Render()
```

#### CHARTTYPE.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```



### Accumulation Charts

Supports 4 different types of accumulation charts as follows,

- Pie
- Doughnut
- Funnel
- Pyramid

As like other chart types it can be changed using the property [Type](#) in [PivotChartSeries](#) class.

In the below code sample, the **Pie** chart is rendered, and the other accumulation charts can be switched using the drop-down list.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
<div class="control-section" style="overflow:auto">
 <div id="dropdown-control" style="margin-bottom:5px;">
 <table style="width: 350px;">
 <tbody>
 <tr style="height: 50px">
 <td>
 <div>
 Chart Type:
 </div>
 </td>
 <td>
 <div>
 <select id="charttypes" name="ddl-view-mode">
 <option value='Pie' selected>Pie</option>
 <option value='Doughnut'>DoughNut</option>
 <option value='Pyramid'>Pyramid</option>
 <option value='Funnel'>Funnel</option>
 </select>
 </div>
 </td>
 </tr>
 </tbody>
 </table>
 </div>
</div>
```

```

 </tbody>
 </table>
</div>
<div class="content-wrapper">
 @Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(f
alse)
 .EnableSorting(true).Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
 })
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product
Categories").Add();
 })
 .ChartSettings(chartSettings =>
 chartSettings.ChartSeries(chartseries
=>chartseries.Type(ChartSeriesType.Pie)))
 .Render();
 </div>
</div>
<script>
 var chartTypesDropDown = new ej.dropdowns.DropDownList({
 floatLabelType: "Auto",
 change: function (args) {
 var pivotObj =
document.getElementById('PivotView').ej2_instances[0];
 pivotObj.chartSettings.chartSeries.type = args.value;
 }
 });
 chartTypesDropDown.appendTo("#charttypes");
</script>

```

### ACCUMULATION.CS

```

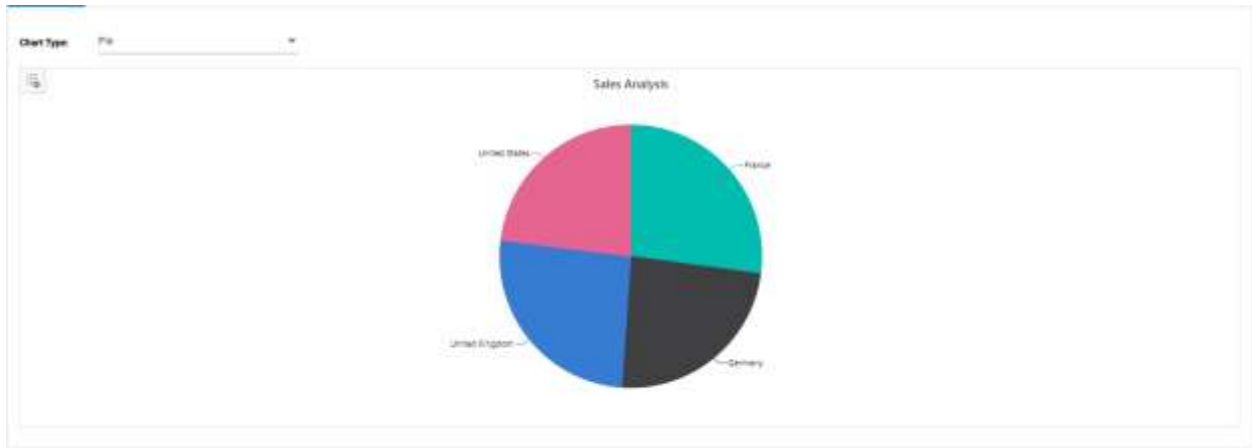
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



```
}

```



### Drill Down/Up

In the accumulation charts, drill down and drill up operations can be performed using the built-in context menu option. It will be shown while clicking on the chart series. The context menu has the following options:

**Expand** - It is to drill down the corresponding series until the last level.

**Collapse** - It is to drill up the corresponding series until the first level.

**Exit** - It is to close the context menu.

**Note:** The drill operation in accumulation charts can be performed only for row headers.

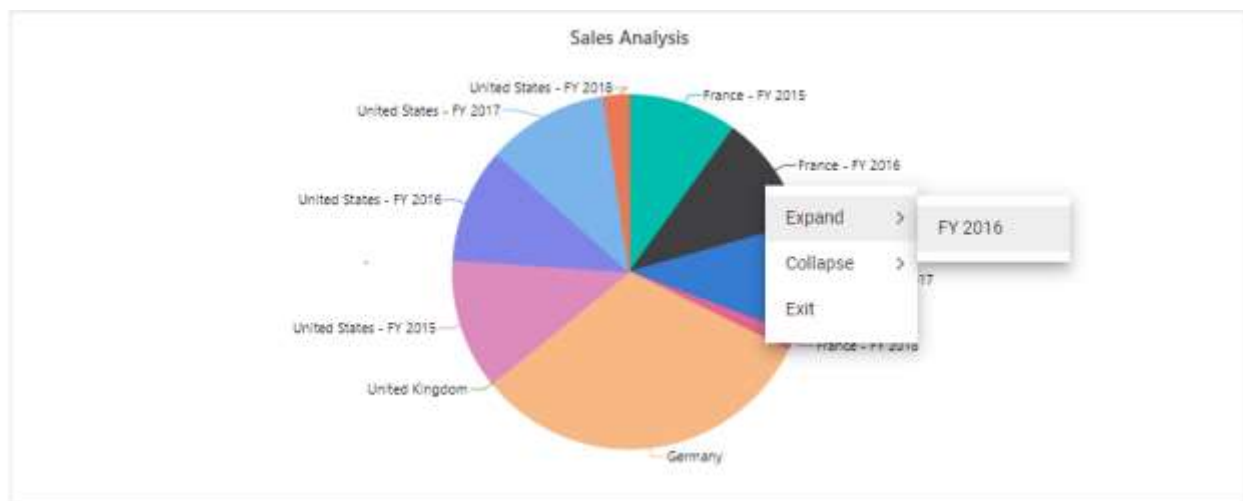
### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.Data).ExpandAll(f
alse)
.EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Year").Add();
 rows.Name("Quarter").Add();
})
.Columns(columns =>
{
 columns.Name("Products").Add();
})
.Values(values =>
{
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C").Add();
})
)
```

```
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
})) .ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries
=> chartSeries.Type(ChartSeriesType.Pie))).Render();
```

### DRILL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



### Column Headers and Delimiters

Unlike other chart types, the accumulation charts consider the values of a single column from the pivot table to be drawn. Preferably the first column of the pivot table is considered by default. But it can be changed by defining the column headers using the `ColumnHeader` property in `ChartSettings` class.

If the column has more than one header, then need to mention all the headers separated by the delimiter -, for example, **FY 2016-Q2**. Using the property `ColumnDelimiter` in `ChartSettings` class, one can set the desired delimiter to separate the column headers.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
).EnableSorting(true).DrilledMembers(drilledmembers =>
{
 drilledmembers.Name("Year").Items(new string[] { "FY 2016" }).Add();
})
).Rows(rows =>
{
```

```

 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
 })
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product Categories").Add();
 })
).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
 {
 chartSeries.Type(ChartSeriesType.Doughnut);
 })
).ColumnHeader("FY 2016-Q2").ColumnDelimiter("-").Render();

```

### COLUMN.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Label Customization

The data labels are visible by default showing header name. Its visibility can be modified using the **Visible** boolean property in **DataLabel**. With regard to the label arrangement, the **Smart Labels** options help to arrange labels efficiently without overlapping. It can be disabled by setting the **EnableSmartLabels** property in **ChartSettings** class as **false**.

The **Position** property in **DataLabel** allows to specify the position of the data label. The available options are,

- **Outside**: Positions the label outside the point. It is the default option.
- **Inside**: Positions the label inside the point.

In the following code sample, the data labels are placed inside.

#### **C#HTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(f
alse)
.EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C").Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries
=>chartSeries.Type(ChartSeriesType.Pyramid).DataLabel(datalabel=>
datalabel.Position("Inside")))).Render();
```

#### **DATALABEL.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.position = Position;
 return View();
}
public PivotViewPivotChartDataLabel Position = new
PivotViewPivotChartDataLabel { Position = "Inside" };
```



The **Connector Line** will be visible when the data label is placed outside the chart. It can be customized using the **ConnectorStyle** property in **DataLabel** for its color, length, width etc. In the following code sample, the connector line is customized.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(f
alse)
.EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C").Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries
=>chartSeries.Type(ChartSeriesType.Funnel).DataLabel(datalabel=>
{
 datalabel.Position("OutSide");

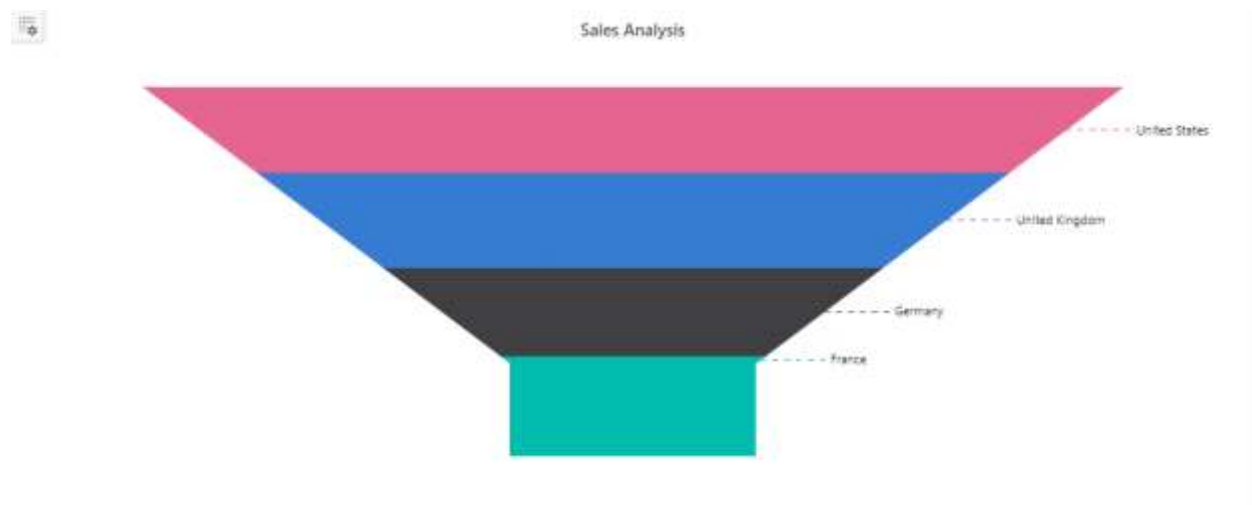
 datalabel.ConnectorStyle(connector=>connector.Color("#f4429e").DashArray("5,
3").Length("50px").Width(2));
})))).Render();
```

**CONNECTOR.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.position = Position;
 return View();
}
public PivotViewPivotChartDataLabel Position = new
PivotViewPivotChartDataLabel { Position = "OutSide" };

```

*Pie and Doughnut Customization*

User can draw pie and doughnut charts within the specified range using the **StartAngle** and **EndAngle** properties in [PivotChartSeries](#) class. The default value of the **StartAngle** property is **0**, and the **EndAngle** property is **360**. By customizing these properties, user can draw semi pie and semi doughnut charts.

**CSHTML**

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
.EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
})
.Values(values =>

```

```

{
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C").Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
{
 chartSeries.Type(ChartSeriesType.Doughnut);
 chartSeries.StartAngle(270);
 chartSeries.EndAngle(90);
})
})
.Render();

```

### ANGLE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



Users can get doughnut chart from pie chart and vice-versa using the `InnerRadius` property in [PivotChartSeries](#) class. If the property is greater than **0** percent, the doughnut chart will appear from the pie chart.

**Note:** It takes the value only in percentage.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(f
alse)

```

```

.EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Amount").Caption("Sales
Amount").Add();values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C").Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
{
 chartSeries.Type(ChartSeriesType.Pie);
 chartSeries.InnerRadius("140");
}
)
).Render();

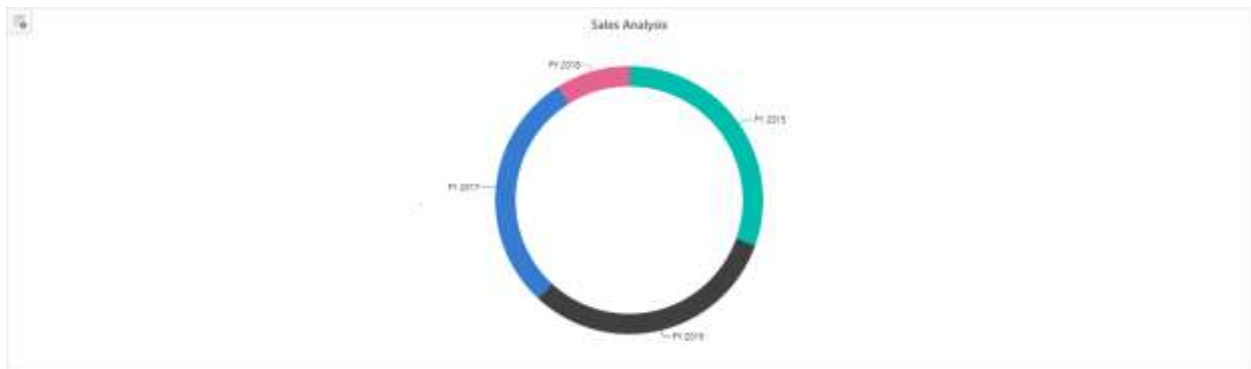
```

### **RADIUS.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



#### *Exploding Series Points*

Exploding can be enabled by setting the **Explode** property in [PivotChartSeries](#) class to **true**. The series points will be exploded either on mouse click or touch.

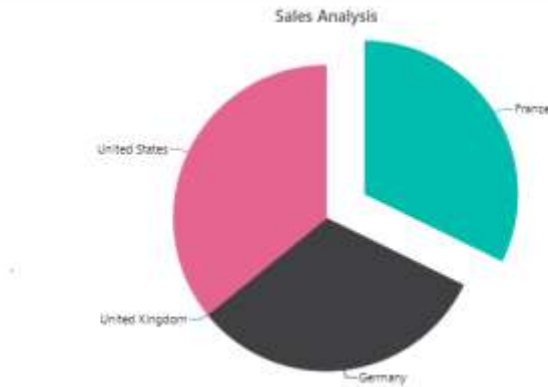


**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.Data).ExpandAll(f
alse)
.EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Amount").Caption("Sales
Amount").Add(); values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C").Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
{
 chartSeries.Type(ChartSeriesType.Pie);
 chartSeries.Explode(true);
})
).Render();
```

**EXPLODE.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



### Field List

User can enable the field list by setting the property [ShowFieldList](#) in [PivotView](#) class as **true**.

By using this, user can customize the report dynamically and view the result in pivot chart. For more information regarding the field list, refer the [field list](#) topic.

In the following sample, the **Popup** mode of field list is enabled in the pivot chart integration.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Column)).ShowFieldList(true).Render()
```

### CHARTFIELDLIST.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

```
}

```



### Grouping Bar

User can enable the grouping bar by setting the property [ShowGroupingBar](#) in [PivotView](#) class as **true**. The grouping bar in pivot chart shows a dropdown list in value axis instead of buttons. The dropdown list holds list of value fields bounded in the [PivotViewDataSourceSettings](#) and it can be switched to draw the pivot chart with the selected value field. This has been defined as the default behavior in the pivot chart component. For more information regarding the grouping bar, refer the [grouping bar](#) topic.

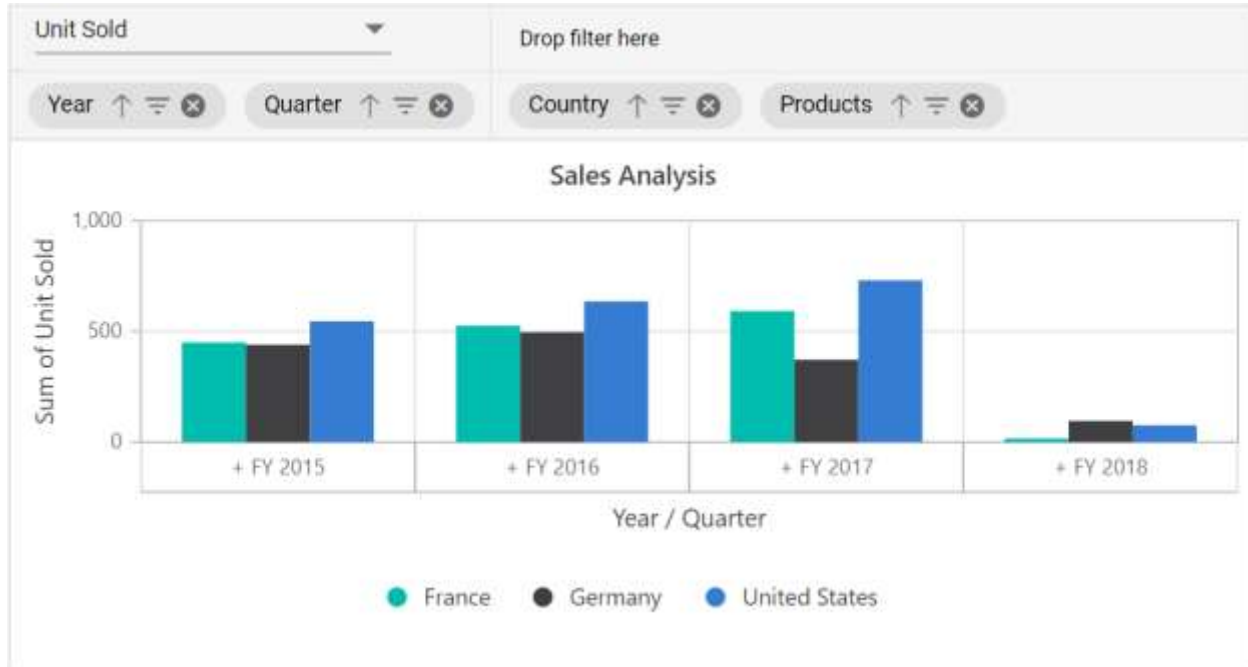
**Note:** For multiple axis support, buttons will be placed in value axis instead of dropdown list.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Column)).ShowGroupingBar(true).Render()
```

**CHARTGROUPINGBAR.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



For accumulation charts alone, a drop-down list will be placed in the column axis instead of the buttons. The drop-down list shows the column headers available in the pivot table. Users can dynamically switch column headers with the help of the drop-down list, and the accumulation chart will be updated accordingly.

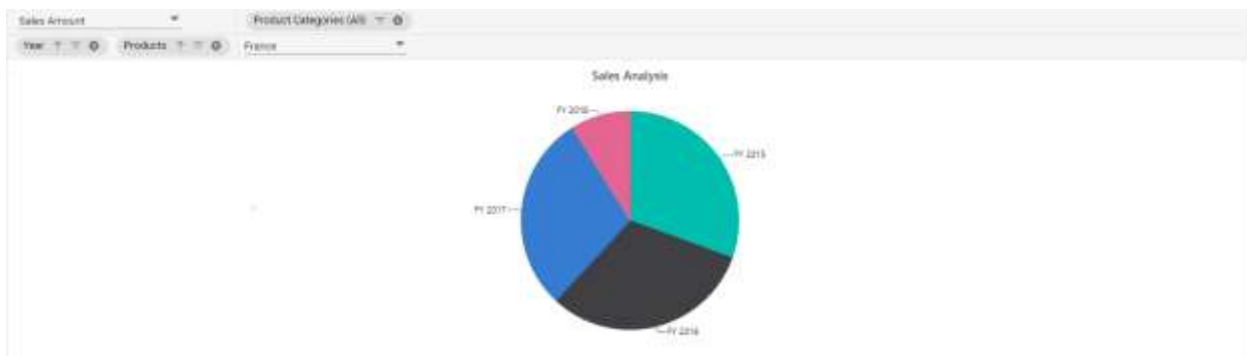
**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Chart
}).ShowFieldList(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
).EnableSorting(true).Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
).Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
})
```

```
.Values(values =>
{
 values.Name("Amount").Caption("Sales
Amount").Add();values.Name("Sold").Caption("Units Sold").Add();
})
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C").Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries
=>chartSeries.Type(ChartSeriesType.Pie)
)).ShowGroupingBar(true).Render();
```

### GROUP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



### Single Axis

By default, the pivot chart will be drawn with the value field (measure) which is set first in the report under value axis. But, user can change to specific value field using the property [Value](#) in [ChartSettings](#) class.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
```

```

 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings => chartSettings.Value("Amount").Title("Sales
 Analysis").ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Column))).Render()

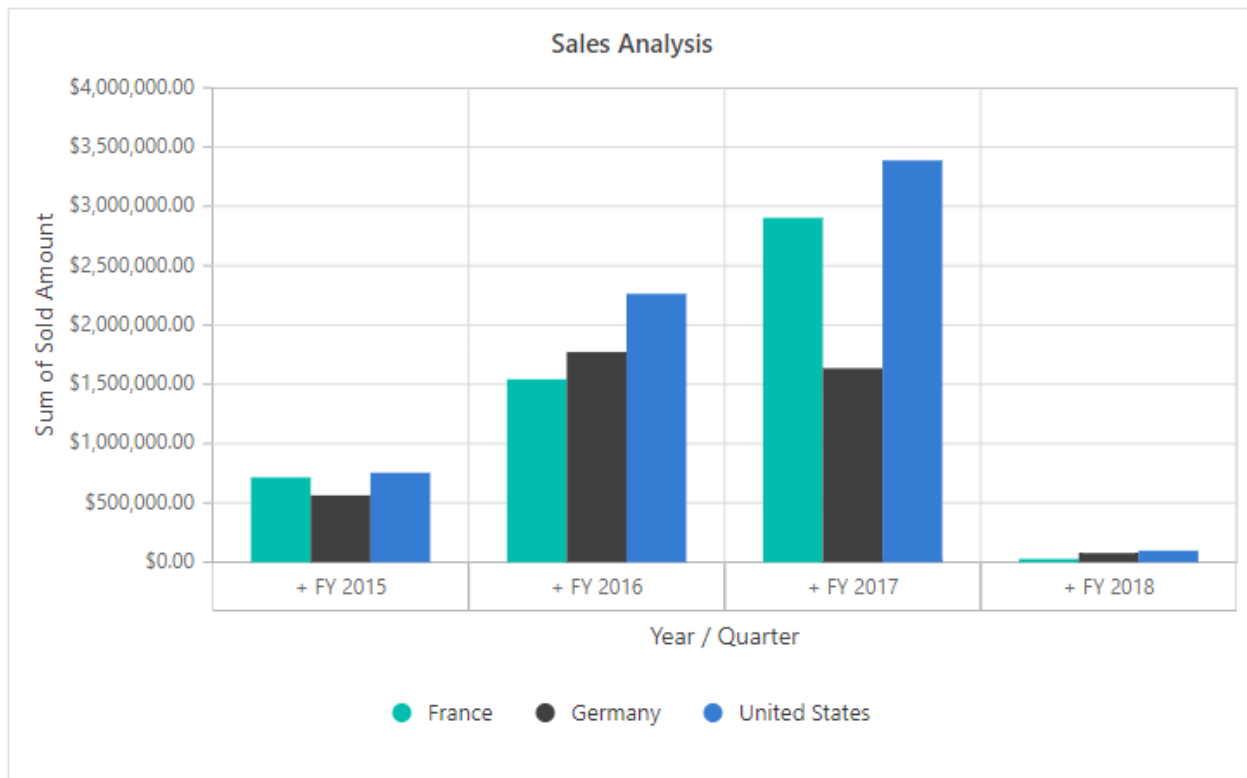
```

### CHARTSINGLE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Multiple Axis

User can draw the pivot chart with multiple value fields by setting the property [EnableMultipleAxis](#) in [ChartSettings](#) class as **true**. In the below code sample, the pivot chart will be drawn with both value fields "Sold" and "Amount" available in the [PivotViewDataSourceSettings](#).

**Note:** The multiple axis support is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings =>
 chartSettings.EnableMultipleAxis(true).ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Column))).Render()
```

### CHARTMULTIVALUE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



If the user binds more value fields, the result will be multiple pivot charts, and each chart will shrink within the parent container height. To avoid this, set the [EnableScrollOnMultiAxis](#) property in [ChartSettings](#) to **true**. By doing so, each pivot chart will only shrink to a minimal "160px" – "180px" height showing a vertical scrollbar for a clear view.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 values.Name("Products").Type("Count").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings =>
```



```
chartSettings.EnableMultipleAxis(true).enableScrollOnMultiAxis(true).ChartSeries(chartSeries => chartSeries.Type(ChartSeriesType.Column)).Render()
```

### CHARTSCROLLBAR.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



Meanwhile, there is another way to display multiple values in a chart. In this approach, the series drawn from multiple values are grouped and displayed in a single chart. And, based on the values, multiple Y axis scales will be framed with different ranges. This can be achieved by setting the properties [EnableMultipleAxis](#) as **true** and [MultipleAxisMode](#) as **Single** in [ChartSettings](#).

In the following code sample, the pivot chart can be seen as a single chart with multiple value fields such as **Sold** and **Amount** that are drawn as multiple Y axis.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}
```

```

 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings =>
 chartSettings.EnableMultipleAxis(true).MultipleAxisMode('Single').ChartSeries
 (chartSeries => chartSeries.Type(ChartSeriesType.Column))).Render()

```

### **CHARTMULTIPLEAXISMODE.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

Additionally, to display chart series for multiple values within a single y-axis, set the properties [EnableMultipleAxis](#) to **true** and the [MultipleAxisMode](#) to **Combined**, in the [ChartSettings](#).

The y-axis range values will be formatted using the first value field on the value axis. For example, if the first value field is in currency format and the remaining value fields are in different number formats or no format, the y-axis range values will be displayed in the currency format of the first value field.

The pivot chart in the following code sample can be seen as a single chart with multiple value fields such as **Sold** and **Amount** drawn as a single y-axis.

### **CSHTML**

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("450").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Production Year").Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
}

```

```

))) .DisplayOption(new PivotViewDisplayOption { View = View.Chart
}) .ChartSettings(chartSettings =>
chartSettings.EnableMultipleAxis(true) .MultipleAxisMode(MultipleAxisMode.Com
bined) .ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column)) .Render()

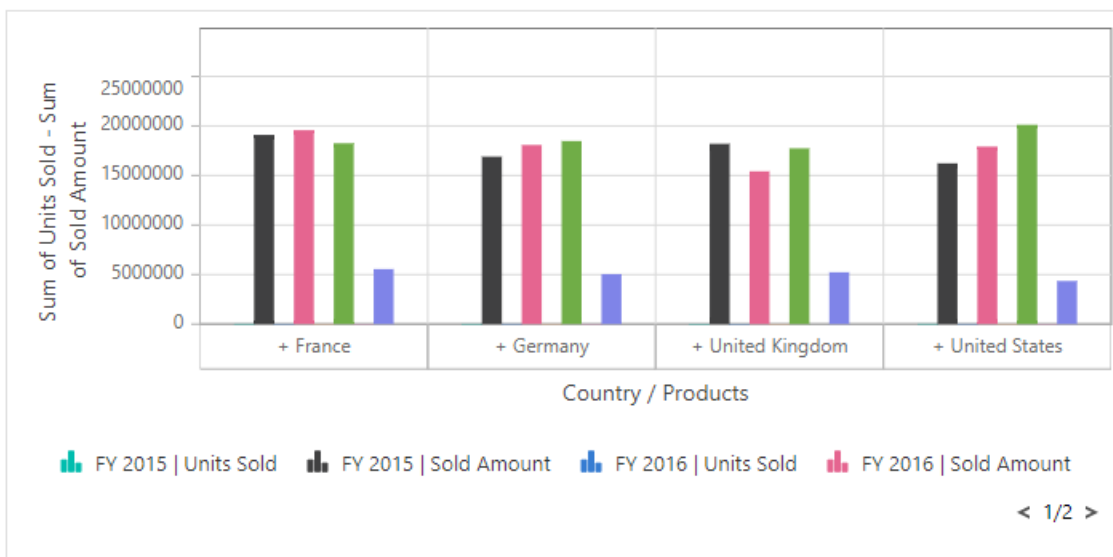
```

### COMBINEDYAXIS.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



#### Show point color based on members

When multiple axes are enabled, you can display the same color for each member in the column axis by setting the [ShowPointColorByMembers](#) property to **true** in the [ChartSettings](#). As a result, the end user can easily identify each member across different measures in the entire chart.

Furthermore, end user can see or hide specific members across different measures in the entire chart with a single click on the legend item.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}

```

```

 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Production Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings =>
 chartSettings.Value("Amount").EnableMultipleAxis(true).MultipleAxisMode(MultipleAxisMode.Stacked).ShowPointColorByMembers(true).ChartSeries(chartSeries => chartSeries.Type(ChartSeriesType.Column)).primaryYAxis(py => py.Border(Width("0")))).Render()

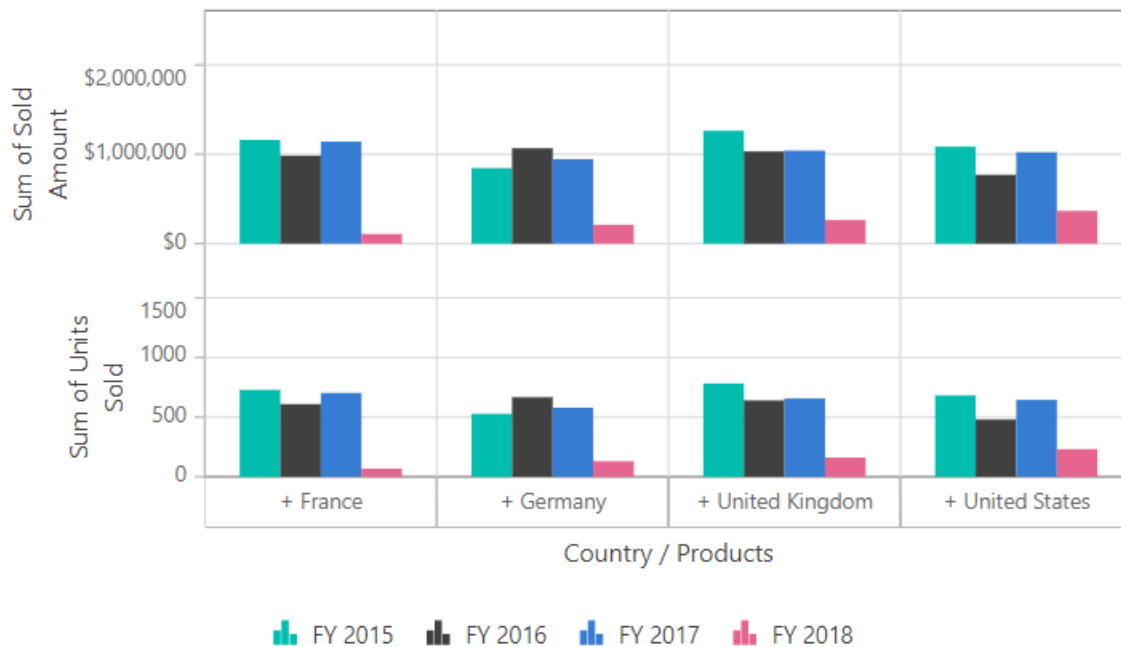
```

### SHOWMEMBERSERIES.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Series Customization

User can customize series of the pivot chart using [ChartSeries](#) in [ChartSettings](#) class. The changes handled in the property will be reflected commonly in all chart series.

In the following sample, the pivot chart type and border has been changed for all the series.

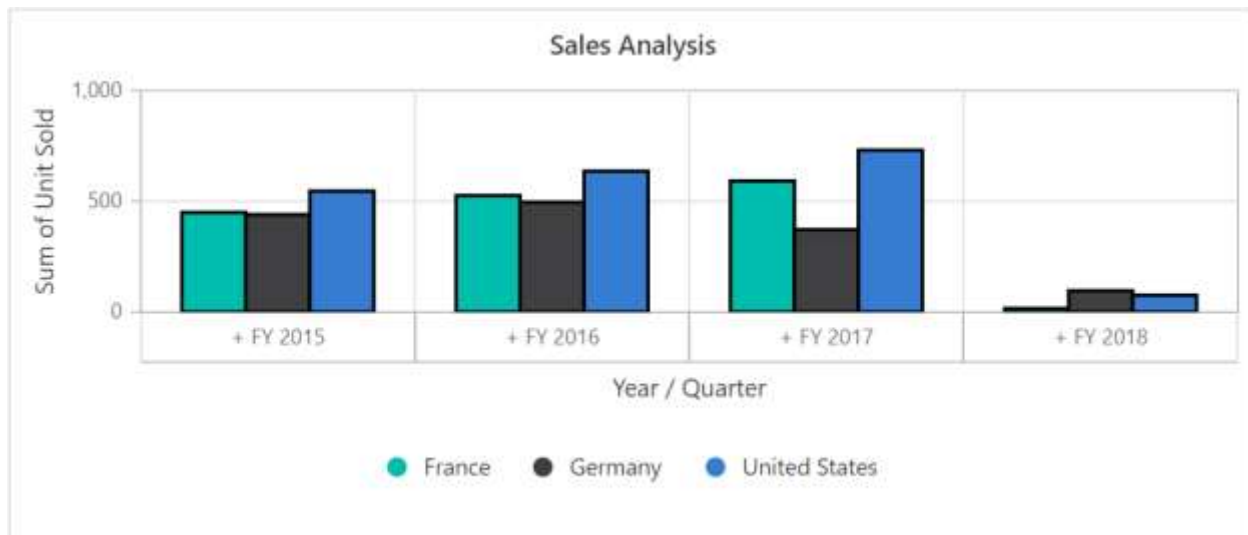
### CSHTML

```
@using Syncfusion.EJ2.PivotView
```

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column).EnableTooltip(false).Border(border
=> border.Color("#000").Width(2))))).Render()
```

### CHARTSERIES.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



User can also customize the pivot chart series individually using the **ChartSeriesCreated** event, which occurs after the pivot chart series has been created. You can customize each series individually by iterating them.

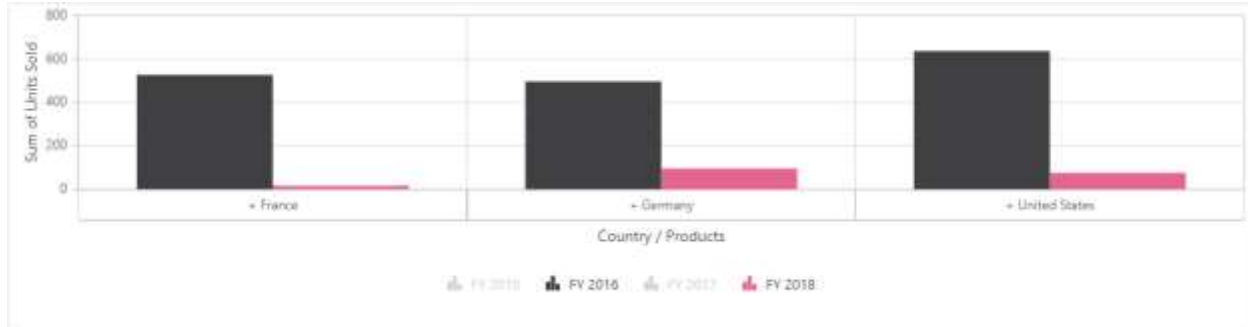
In the following sample, the even series are hidden in the pivot chart.

### CSSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ChartSeriesCreated("chartSeriesCreated").DisplayOption(new
PivotViewDisplayOption { View = View.Chart }).ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()
<script>
function chartSeriesCreated(args) {
 for (var pos = 0; pos < args.series.length; pos++) {
 if (pos % 2 == 0) {
 args.series[pos].visible = false;
 }
 }
}
</script>
```

### CHARTSERIESEVENT.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



### Axis Customization

User can customize axis of the pivot chart using [PrimaryXAxis](#) and [PrimaryYAxis](#) properties in [ChartSettings](#) class.

**Note:** Axis customization is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, title of y-axis and x-axis are customized.

### CSHTML

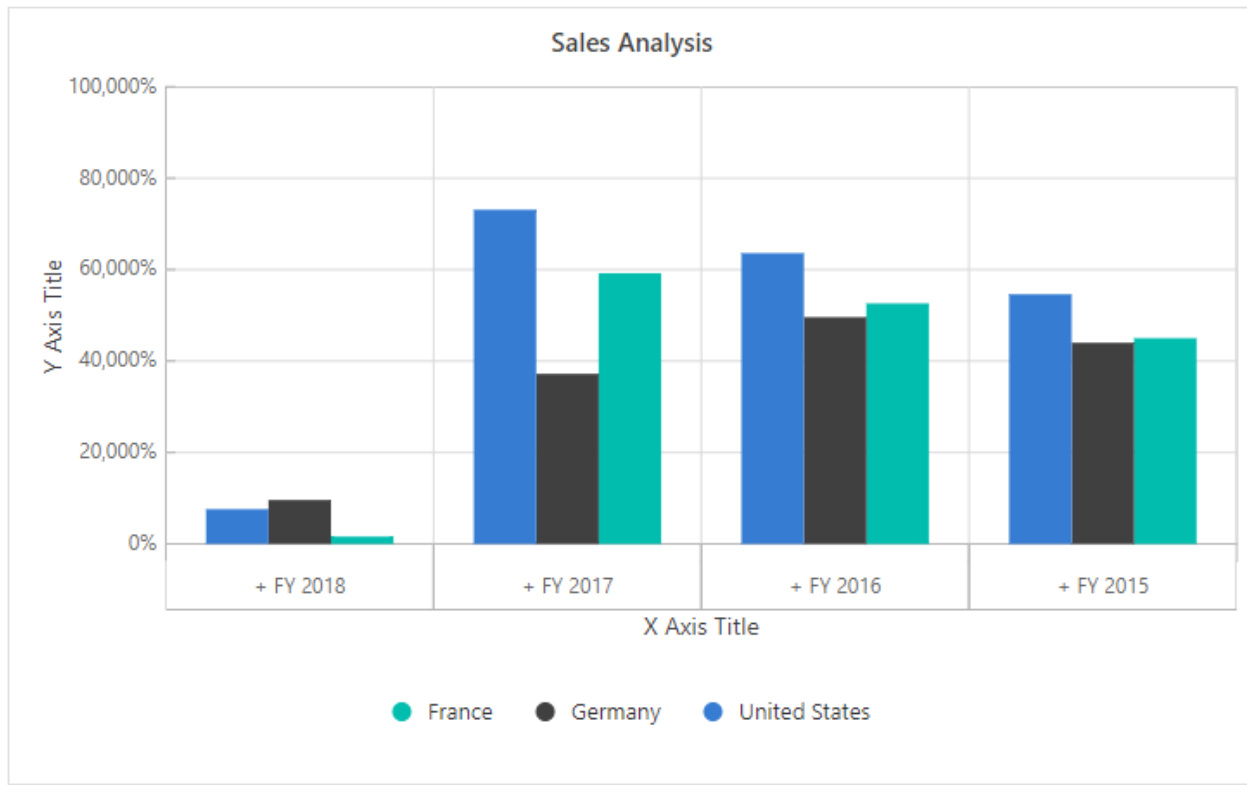
```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column)).PrimaryXAxis(primaryXAxis =>
primaryXAxis.Title("X axis title")).PrimaryYAxis(primaryYAxis =>
primaryYAxis.Title("Y axis title"))).Render()
```

### CHARTAXIS.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

```
}

```



One can also customize multi-level labels of primary x-axis by using the `MultiLevelLabelRender` event in the [ChartSettings](#), which fires on rendering each multi-level label in the pivot chart. It has the following parameters:

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}
```



```

 }).MultiLevelLabelRender("multiLevelLabelRender").DisplayOption(new
 PivotViewDisplayOption { View = View.Chart }).ChartSettings(chartSettings =>
 chartSettings.ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Column)).Render()
<script>
 function multiLevelLabelRender(args) {
 args.alignment = 'Far';
 args.textStyle = { fontFamily: 'Bold', fontWeight: '400', size:
 '16px', color: 'red' };
 if (args.text === ' + United States') {
 args.text = 'Text Changed';
 args.alignment = 'Near';
 args.textStyle = { fontFamily: 'Bold', fontWeight: '800', size:
 '16px', color: 'Blue' };
 }
 }
</script>

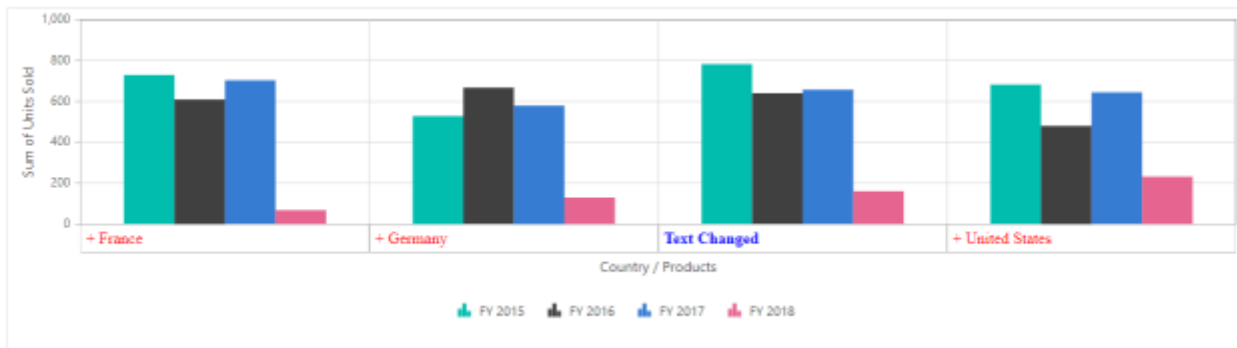
```

### CHART-MULTILEVELLABELRENDER.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Legend Customization

User can customize legend using [LegendSettings](#) in [ChartSettings](#) class. By default, legend will be visible and it can be hidden by setting the property [Visible](#) in [LegendSettings](#) class as **false**.

The pivot chart support different types of legend shapes as follows,

- Circle
- Rectangle
- VerticalLine
- Pentagon
- InvertedTriangle
- SeriesType
- Triangle
- Diamond

- Cross
- HorizontalLine

Here **SeriesType** would act as the default shape and it can be changed using the property [LegendShape](#) in [ChartSeries](#) class.

Also user can set the position of the legend in pivot chart using the property [Position](#) in [LegendSettings](#) class. The available options to set the legend position are as follows,

- Auto: Places the legend based on area type. This is the default.
- Top: Displays the legend at the top of the pivot chart.
- Left: Displays the legend at the left of the pivot chart.
- Bottom: Displays the legend at the bottom of the pivot chart.
- Right: Displays the legend at the right of the pivot chart.
- Custom: Displays the legend based on the given x and y values.

**Note:** By default, the legend is not visible for the accumulation chart types like pie, doughnut, pyramid, and funnel.

In the following sample, the legend shape and its position can be customized.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column).LegendShape("Pentagon")).LegendSettings(legendSettings => legendSettings.Position("Right"))).Render()
```

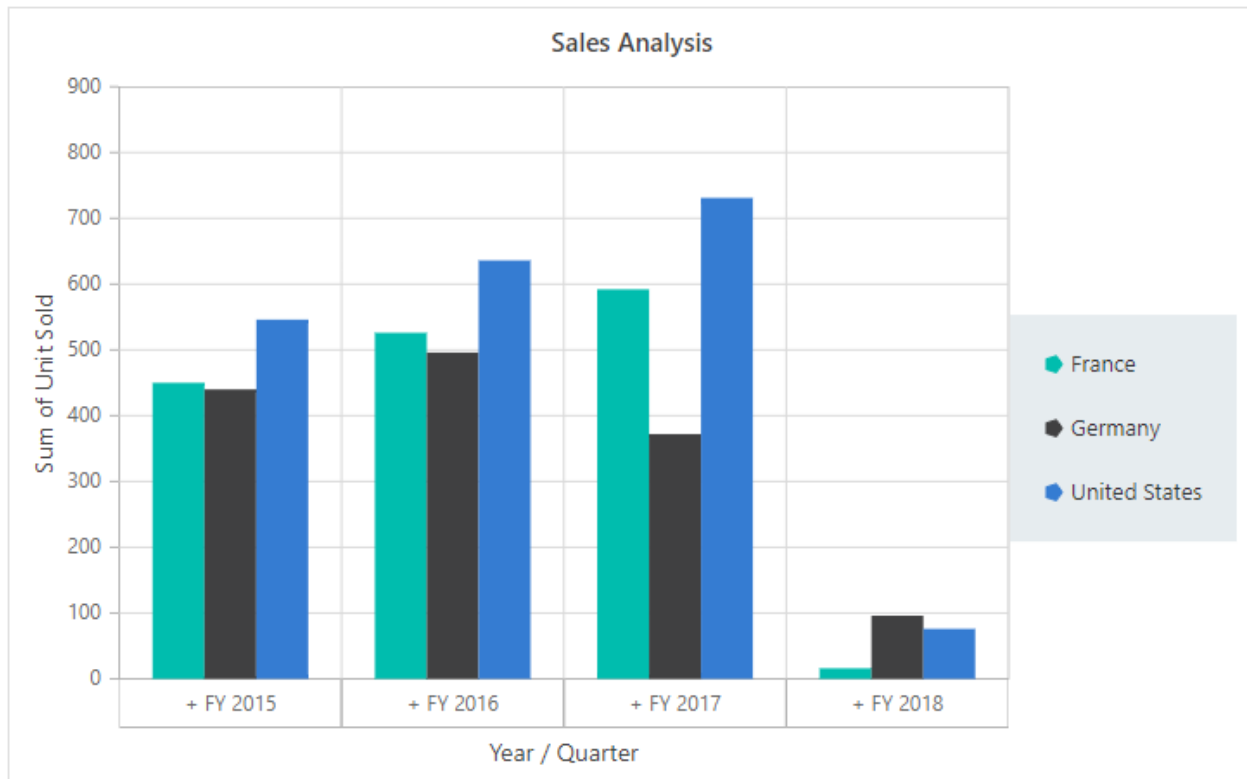
#### LEGEND.CS

```
public ActionResult Index()
{
var data = GetPivotData();
```

```

 ViewBag.DataSource = data;
 return View();
}

```



### User Interaction

#### Marker and CrossHair

User can enable and customize the marker and crosshair using [MarkerSettings](#) and [CrosshairSettings](#) properties in [ChartSettings](#) class respectively.

Also user can enable and customize the crosshair tooltip for axes using [PrimaryXAxis](#) and [PrimaryYAxis](#) classes.

**Note:** Marker and crosshair is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{

```

```

 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Line).Marker(marker =>
 marker.Fill("#EEE").Height(10).Width(10).Shape("Pentagon").Visible(true)).Crosshair(crosshair => crosshair.Enable(true)).PrimaryXAxis(primaryXAxis =>
 primaryXAxis.CrosshairTooltip(crosshairTooltip =>
 crosshairTooltip.Enable(true).Fill("#ff0000")).PrimaryYAxis(primaryYAxis =>
 primaryYAxis.CrosshairTooltip(crosshairTooltip =>
 crosshairTooltip.Enable(true).Fill("#0000FF")))).Render()

```

### MARKER.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Zooming and Panning

User can customize zooming and panning option using the property [ChartZoomSettings](#) in [ChartSettings](#) class.

The pivot chart support four types of zooming which can be set as follows,

- [EnablePinchZooming](#)
- [EnableSelectionZooming](#)

- [EnableDeferredZooming](#)
- [EnableMouseWheelZooming](#)

and three modes of zooming direction that specifies whether to zoom vertically or horizontally or in both ways which are,

- x: Pivot chart can be zoomed horizontally.
- y: Pivot chart can be zoomed vertically.
- x,y: Pivot chart can be zoomed both vertically and horizontally.

This can be set using the property [Mode](#) in [ZoomSettings](#) class. By default, if the pivot chart is zoomed, a toolbar would display with the options - Zoom, ZoomIn, ZoomOut, Pan, Reset. User can also customize its option using the property [ToolbarItems](#) in [ZoomSettings](#) class.

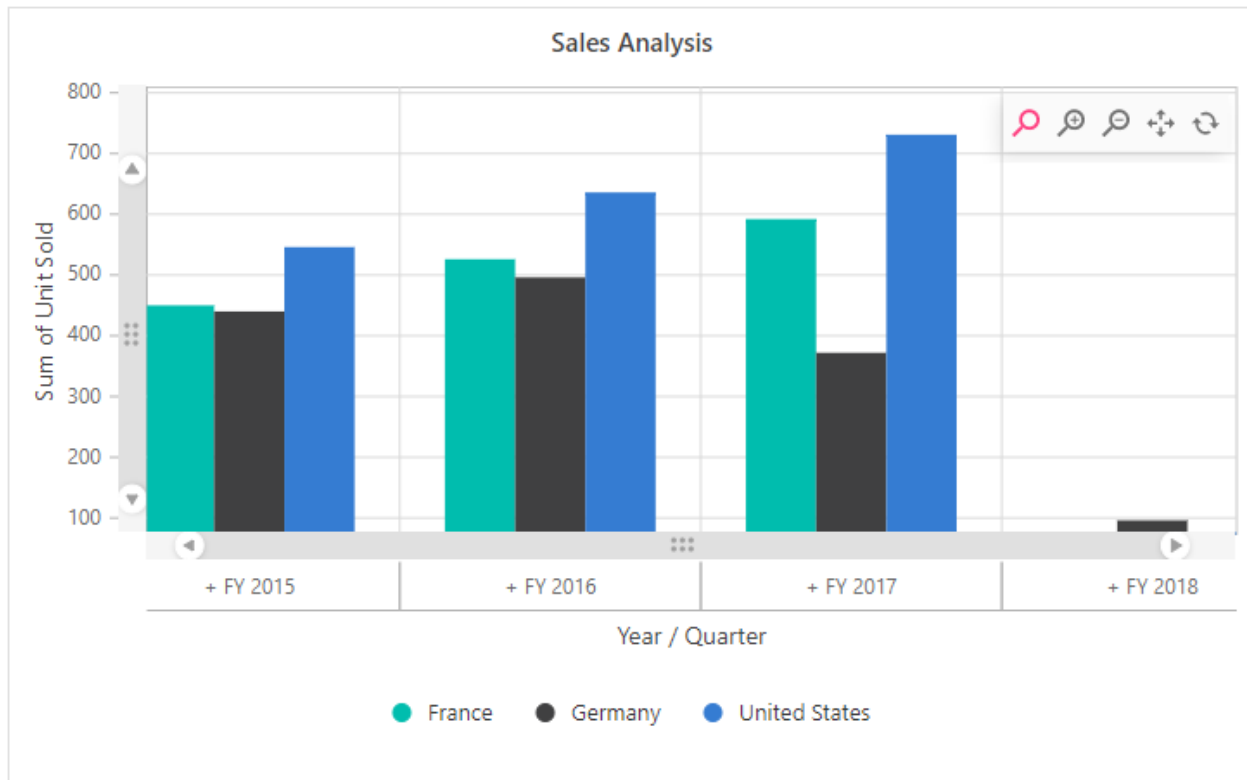
**Note:** Zooming and panning is not applicable for the accumulation chart types like pie, doughnut, pyramid, and funnel.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column)).ZoomSettings(zoomSettings =>
zoomSettings.EnableDeferredZooming(true).EnableMouseWheelZooming(true).EnablePinchZooming(true).EnableSelectionZooming(true))).Render()
```

### ZOOMING.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```



### Tooltip

By default, tooltip for the pivot chart is enabled. User can customize it by using the property [TooltipSettings](#) in [ChartSettings](#) class.

**Note:** The tooltip can be disabled by setting the property [Enable](#) in [TooltipSettings](#) class as **false**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}
```

```

 }).DisplayOption(new PivotViewDisplayOption { View = View.Chart
 }).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
 chartSeries.Type(ChartSeriesType.Column)).Tooltip(tooltip =>
 tooltip.EnableMarker(true).Fill("#FFF").Opacity(1).TextStyle(textStyle =>
 textStyle.Color("#000")).Border(border => border.Color("#000")))).Render()

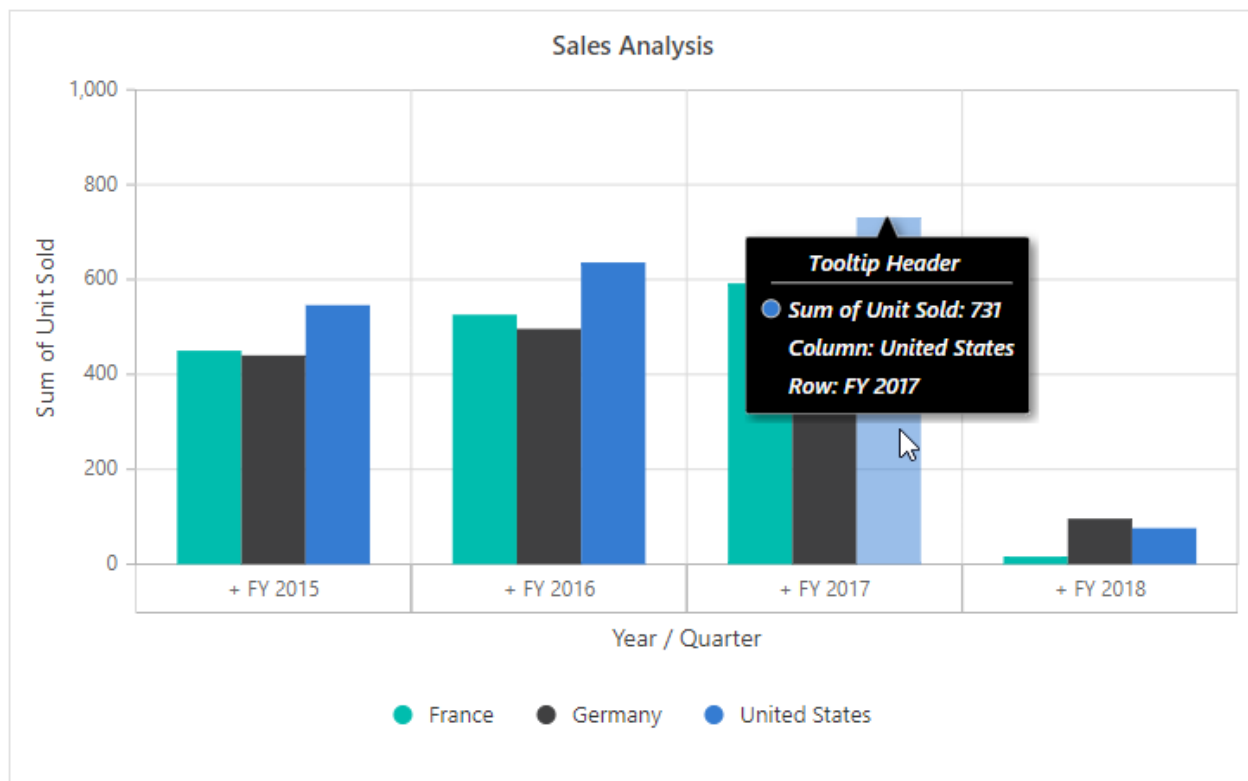
```

### CHARTTOOLTIP.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Export

The pivot chart can be exported using the `chartExport` method which holds parameters like export type, file name, PDF orientation, width, and height in the same order. The mandatory parameters for this method are export type and file name whereas other parameters are optional.

The following are the four export types:

- PNG
- JPEG
- SVG
- PDF

In the following code sample, exporting can be done using an external button named as "Chart Export".

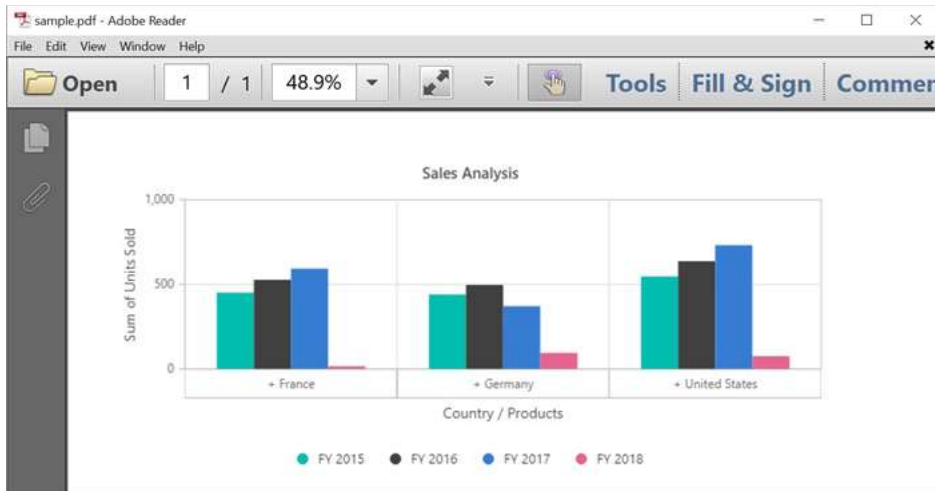
### CHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("chartexport").Content("Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()
<script>
var pivotObj;
document.getElementById("chartexport").onclick = function () {
pivotObj = document.getElementById("PivotView").ej2_instances[0];
pivotObj.chartExport("PNG", "result");
}
</script>
```

### EXPORT.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```





### Print

The rendered pivot chart can be printed directly from the browser by calling `printChart` method.

In the following code sample, printing can be done using an external button named as "Print".

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("chartprint").Content("Print").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()
<script>
var pivotObj;
document.getElementById("chartprint").onclick = function () {
pivotObj = document.getElementById("PivotView").ej2_instances[0];
pivotObj.printChart();
}
</script>
```

**PRINT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**Drill Down in ASP.NET MVC Pivot Table Control****Drill down and drill up**

The drill down and drill up action helps to view the bound data in detailed and abstract view respectively. By default, if member(s) has children, then expand and collapse icon will be displayed in the respective row/column header. On clicking the icon, expand or collapse action will be performed automatically through built-in source code. Meanwhile, leaf member(s) does not contain expand and collapse icon.

|                 | ► FY 2015  |             | ► FY 2016  |             | ► FY 2017  |
|-----------------|------------|-------------|------------|-------------|------------|
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▼ France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Mountain Bikes  | 197        | \$335,688   | 238        | \$405,552   |            |
| Road Bikes      | 253        | \$379,267   | 288        | \$1,136,552 |            |
| ► Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| ► United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total     | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

**Drill position**

Allows to drill only the current position of the selected member and exclude the drilled data of selected member in other positions. For example, if "FY 2015" and "FY 2016" have "Q1" member as child in next level, and when end user attempts to drill "Q1" under "FY 2016", only it will be expanded and not "Q1" under "FY 2015".

**Note:** This feature is built-in and occurs every time when expand or collapse action is done for better performance.

|               | FY 2015 |                | FY 2016        |            |          |
|---------------|---------|----------------|----------------|------------|----------|
|               | Q1      | FY 2015 Tot... | Q1             |            |          |
|               |         |                | Mountain Bi... | Road Bikes | Q1 Total |
| France        | 114     | 114            | 27             | 67         |          |
| Germany       | 74      | 74             | 97             | 57         |          |
| United States | 144     | 144            | 57             | 97         |          |
| Grand Total   | 332     | 332            | 181            | 221        |          |

### Expand all

**Note:** This property is applicable only for the relational data source.

Allows to either expand or collapse all headers that are displayed in row and column axes. To display all headers in expanded state, set the property [ExpandAll](#) to **true** and to collapse all headers, set the property [ExpandAll](#) to **false**. By default, [ExpandAll](#) property is set to **false**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### EXPANDALL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                 | ▼ FY 2015  |             |            |             |            |
|-----------------|------------|-------------|------------|-------------|------------|
|                 | Q1         |             | Q2         |             | Q3         |
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▼ France        | 114        | \$177,246   | 108        | \$172,352   |            |
| Mountain Bikes  | 31         | \$52,824    | 51         | \$86,904    |            |
| Road Bikes      | 83         | \$124,422   | 57         | \$85,448    |            |
| ▼ Germany       | 74         | \$117,246   | 128        | \$152,352   |            |
| Mountain Bikes  | 51         | \$92,824    | 61         | \$76,904    |            |
| Road Bikes      | 23         | \$24,422    | 67         | \$75,448    |            |
| ▼ United States | 144        | \$162,246   | 171        | \$145,352   |            |
| Mountain Bikes  | 91         | \$67,824    | 81         | \$99,904    |            |

Expand all headers for specific fields

**Note:** This property is applicable only for the relational data source.

Allows to expand or collapse all headers for specific fields (only) in row and column axes. To expand headers for a specific field in row or column axis, set the property [ExpandAll](#) in [Rows](#) or [Columns](#) to **true**. By default, [ExpandAll](#) property in [Rows](#) or [Columns](#) is set to **false**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.Rows(rows =>
{
 rows.Name("Country").ExpandAll(true).Add();
 rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").ExpandAll(true).Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### EXPANDALL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                 | ▼ FY 2015  |             |            |             |            |
|-----------------|------------|-------------|------------|-------------|------------|
|                 | Q1         |             | Q2         |             | Q3         |
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▼ France        | 114        | \$177,246   | 108        | \$172,352   |            |
| Mountain Bikes  | 31         | \$52,824    | 51         | \$86,904    |            |
| Road Bikes      | 83         | \$124,422   | 57         | \$85,448    |            |
| ▼ Germany       | 74         | \$117,246   | 128        | \$152,352   |            |
| Mountain Bikes  | 51         | \$92,824    | 61         | \$76,904    |            |
| Road Bikes      | 23         | \$24,422    | 67         | \$75,448    |            |
| ▼ United States | 144        | \$162,246   | 171        | \$145,352   |            |
| Mountain Bikes  | 91         | \$67,824    | 81         | \$99,904    |            |

Expand all except specific member(s)

**Note:** This option is applicable only for the relational data source.

In addition to the previous topic, there is an enhancement to expand all headers except specific header(s) and similarly to collapse all headers except specific header(s). To achieve this, [PivotViewDrilledMember](#) class is used. The required properties of the [PivotViewDrilledMember](#) class are explained below:

- [Name](#): It allows to set the field name whose member(s) needs to be specifically drilled.
- [Items](#): It allows to set the exact member(s) which needs to be drilled.

**Note:** The [PivotViewDrilledMember](#) option always works in vice-versa with respect to the property [ExpandAll](#) in pivot table. For example, if [ExpandAll](#) is set to **true**, then the member(s) added in [Items](#) collection alone will be in collapsed state.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
.DrilledMembers(drilledmembers =>
{
 drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).Render()

```

### DRILLED MEMBERS.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.drilledMembers = new string[] { "France", "Germany" };
 return View();
}

```

|                 | FY 2015    |             |            |             |            |
|-----------------|------------|-------------|------------|-------------|------------|
|                 | Q1         |             | Q2         |             | Q3         |
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▶ France        | 114        | \$177,246   | 108        | \$172,352   | 1          |
| ▶ Germany       | 74         | \$117,246   | 128        | \$152,352   | 1          |
| ▼ United States | 144        | \$162,246   | 171        | \$145,352   |            |
| Mountain Bikes  | 91         | \$67,824    | 81         | \$99,904    |            |
| Road Bikes      | 53         | \$94,422    | 90         | \$45,448    |            |
| Grand Total     | 332        | \$456,738   | 407        | \$470,056   | 3          |

### Expand specific member(s)

End user can also manually expand or collapse specific member(s) in each fields under row and column axes using the [DrilledMembers](#) class from code behind. The required properties of the [DrilledMembers](#) class are explained below:

- [Name](#): It allows to set the field name whose member(s) needs to be specifically drilled.
- [Items](#): It allows to set the exact member(s) which needs to be drilled.
- [Delimiter](#): It allows to separate next level of member from its parent member.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .DrilledMembers(drilledmembers =>
 {
 drilledmembers.Name("Quarter").Delimiter("~").Items(ViewBag.drilledMem).Add();
 drilledmembers.Name("Year").Items(ViewBag.drilledMembers).Add();
 })
 .Rows(rows =>
 {

```

```

 rows.Name("Year").Caption("Year").Add(); rows.Name("Quarter").Add();
 rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Country").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()

```

### DRILLED MEMBERS.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.drilledMembers = new string[] { "FY 2015", "FY 2016" };
 ViewBag.drilledMem = new string[] { "FY 2015~~Q1" };
 return View();
}

```

|                | France     |             | Germany    |             | United St |
|----------------|------------|-------------|------------|-------------|-----------|
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sol |
| ▼ FY 2015      | 450        | \$714,955   | 440        | \$563,515   |           |
| ▼ Q1           | 114        | \$177,246   | 74         | \$117,246   |           |
| Mountain Bikes | 31         | \$52,824    | 51         | \$92,824    |           |
| Road Bikes     | 83         | \$124,422   | 23         | \$24,422    |           |
| ▶ Q2           | 108        | \$172,352   | 128        | \$152,352   |           |
| ▶ Q3           | 110        | \$183,345   | 140        | \$95,705    |           |
| ▶ Q4           | 118        | \$182,012   | 98         | \$198,212   |           |
| ▼ FY 2016      | 526        | \$1,542,104 | 496        | \$1,772,104 |           |
| ▶ Q1           | 94         | \$116,016   | 154        | \$176,016   |           |
| ▶ Q2           | 138        | \$143,992   | 98         | \$183,992   |           |
| ▶ Q3           | 170        | \$963,760   | 90         | \$603,760   |           |

### Event

#### Drill

The event **Drill** triggers every time when a field is expanded or collapsed. For instance using this event user can alter delimiter and drill action for the respective item. It has the following parameters:

- **drillInfo** - It holds the current drilled item information.
- **pivotview** - It holds pivot table instance.

**CSHTML**

```
@Html.EJS().PivotView("pivotview").Width("100%").Height("400").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>
) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
 rows.Name("Country").Add();
 rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
})
.Filters(filters =>
{
 filters.Name("Product_Categories").Caption("Product Categories").Add();
})).Drill("onDrill").Render()
<script>
function onDrill(args) {
 // triggers whenever a cell is expanded or collapsed
}
</script>
```

**DRILLEVENT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.drilledMembers = new string[] { "France", "Germany" };
 return View();
}
```

*ActionBegin*

The event [actionBegin](#) triggers when the UI actions such as drill down and drill up begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.



- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

| Action | Action Name |

|-----|-----|

| [Expand](#) | Drill down |

| [Collapse](#) | Drill up |

- **cancel**: It allows user to restrict the current action.

In the below sample, drill down and drill up action can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Drill down' || args.actionName == 'Drill
up') {
 args.cancel = true;
 }
 }
</script>
```

### ACTIONBEGIN-DRILL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionComplete

The event [actionComplete](#) triggers when a UI action such as drill down or drill up, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action                   | Action Name |
|--------------------------|-------------|
| <a href="#">Expand</a>   | Drill down  |
| <a href="#">Collapse</a> | Drill up    |

- **actionInfo**: It holds the unique information about the current UI action. For example, if drill down action is completed, the event argument contains information such as field name and the drill information.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = true
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'Drill down' || args.actionName == 'Drill
up') {
 // Triggers when the drill operations are completed.
 }
 }
</script>
```

### ACTIONCOMPLETE-DRILL.CS

```
public ActionResult Index()
{
```

```

var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```

### ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

| Action                   | Action Name |
|--------------------------|-------------|
| -----                    | -----       |
| <a href="#">Expand</a>   | Drill down  |
| <a href="#">Collapse</a> | Drill up    |

- **errorInfo**: It holds the error information of the current UI action.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Drill down' || args.actionName == 'Drill
up') {
 // Triggers when the current UI action fails to achieve the
desired result.
 }
 }
}
</script>

```

### ACTIONFAILURE-DRILL.CS

```

public ActionResult Index()

```

```
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

## Data Shaping

### Aggregation

**Note:** This feature is applicable only for relational data source.

End user can perform calculations over a group of values (exclusively for value fields bound in value axis) using the aggregation option. By default, values are added (summed) together. The other aggregation types are explained below.

**Note:** The fields with data type such as number support all aggregation types mentioned below except for **“CalculatedField”**. The fields with data type such as string, date, datetime, boolean, etc., support **“Count”** and **“DistinctCount”** aggregation types alone.

| Operator                   | Description                                                                                                   |
|----------------------------|---------------------------------------------------------------------------------------------------------------|
| ----- -----                |                                                                                                               |
| Sum                        | Displays the pivot table values with sum.                                                                     |
| Product                    | Displays the pivot table values with product.                                                                 |
| Count                      | Displays the pivot table values with count.                                                                   |
| DistinctCount              | Displays the pivot table values with distinct count.                                                          |
| Min                        | Displays the pivot table with minimum value.                                                                  |
| Max                        | Displays the pivot table with maximum value.                                                                  |
| Avg                        | Displays the pivot table values with average.                                                                 |
| Median                     | Displays the pivot table values with median.                                                                  |
| Index                      | Displays the pivot table values with index.                                                                   |
| PopulationStDev            | Displays the pivot table values with standard deviation of population.                                        |
| SampleStDev                | Displays the pivot table values with sample standard deviation.                                               |
| PopulationVar              | Displays the pivot table values with variance of population.                                                  |
| SampleVar                  | Displays the pivot table values with sample variance.                                                         |
| RunningTotals              | Displays the pivot table values with running totals.                                                          |
| DifferenceFrom             | Displays the pivot table values with difference from the value of the base item in the base field.            |
| PercentageOfDifferenceFrom | Displays the pivot table values with percentage difference from the value of the base item in the base field. |
| PercentageOfGrandTotal     | Displays the pivot table values with percentage of grand total of all values.                                 |
| PercentageOfColumnTotal    | Displays the pivot table values in each column with percentage of total values for the column.                |

| [PercentageOfRowTotal](#) | Displays the pivot table values in each row with percentage of total values for the row. |

| [PercentageOfParentTotal](#) | Displays the pivot table values with percentage of total of all values based on selected field. |

| [PercentageOfParentColumnTotal](#) | Displays the pivot table values with percentage of its parent total in each column. |

| [PercentageOfParentRowTotal](#) | Displays the pivot table values with percentage of its parent total in each row. |

| [CalculatedField](#) | Displays the pivot table with calculated field values. It allows user to create a new calculated field alone. |

#### *Assigning aggregation type for value fields through API*

For each value field, the aggregation type can be set using the property [Type](#) in [Value](#) class. Meanwhile, aggregation types like [SummaryTypes.DifferenceFrom](#) and [SummaryTypes.PercentageOfDifferenceFrom](#) can check for specific field of specific item using [BaseField](#) and [BaseItem](#) properties. Likewise, [SummaryTypes.PercentageOfParentTotal](#) type can for specific field using [BaseField](#) property. For instance, the aggregation type [SummaryTypes.DifferenceFrom](#) would intake the specified field and its corresponding member as input and its value is compared across other members in the same field and also across different fields to formulate an appropriate output value.

- [Type](#): It allows to set the aggregate type of the field.
- [BaseField](#): It allows to set the specific field to aggregate the values.
- [BaseItem](#): It allows to set the specific member to aggregate the values.

### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Type(Syncfusion.EJ2.PivotView.SummaryTypes.DifferenceFrom).BaseField("Country").BaseItem("France").Add();
 values.Name("Amount").Caption("Sold Amount").Type(Syncfusion.EJ2.PivotView.SummaryTypes.Min).Add();
 })).Render()
```

### **AGGREGATION.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

}

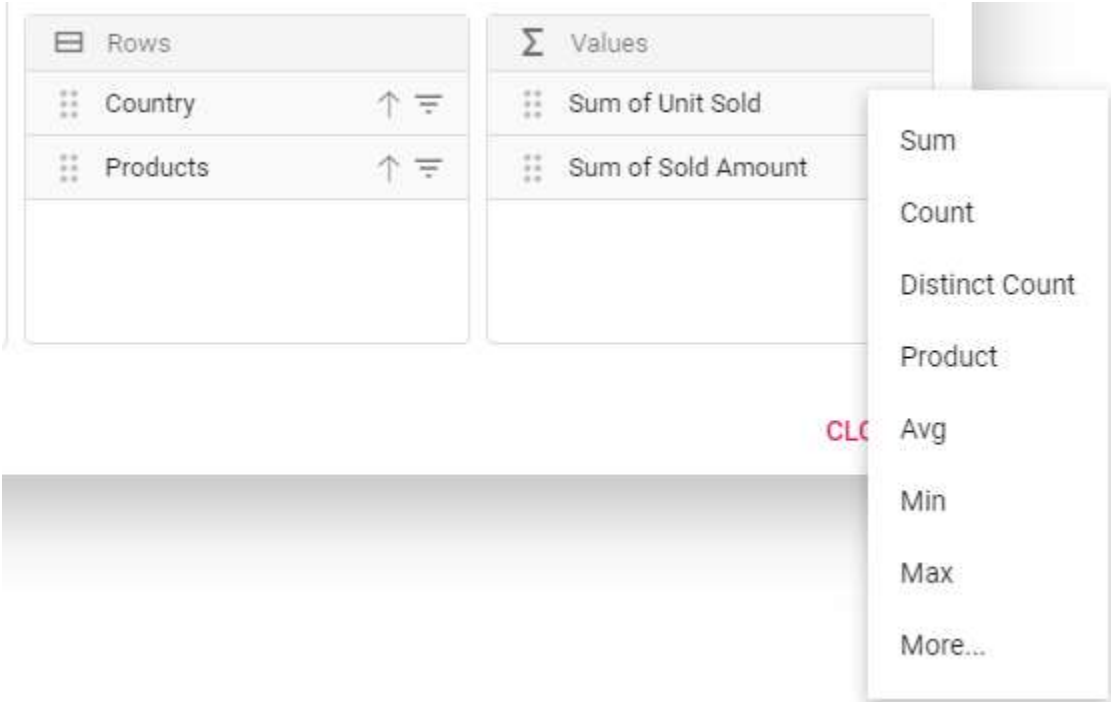
|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        |            | \$29,985    |            | \$46,008    |            |
| Germany       | -10        | \$24,422    | -30        | \$86,008    |            |
| United States | 96         | \$45,448    | 110        | \$90,008    |            |
| Grand Total   |            | \$24,422    |            | \$46,008    |            |

**Note:** By default, the aggregation will be considered as [SummaryTypes.Sum](#) to the value fields which had number type and for the value fields which had non-number type values such as string, date, datetime, boolean, etc., the aggregation type will be considered as [SummaryTypes.Count](#).

*Modifying aggregation type for value fields at runtime*

Aggregation types can be changed easily through UI at runtime. The value fields bound to grouping bar and field list appears with a dropdown icon which helps to select an appropriate aggregation type for the respective value field. On selection, the values in the pivot table will be changed dynamically.

<!-- markdownlint-disable MD012 -->



<br/>

<br/>

| Drop filter here |  | FY 2015    |             | FY 2016    |
|------------------|--|------------|-------------|------------|
|                  |  | Units Sold | Sold Amount | Units Sold |
| France           |  | 450        | \$714,955   | 526        |
| Germany          |  | 440        | \$563,515   | 496        |
| United States    |  | 546        | \$754,515   | 636        |
| Grand Total      |  | 1436       | \$2,032,985 | 1658       |

*Show desired aggregation types in its dropdown menu*

By default, all the aggregation types are displayed in the dropdown menu available in buttons. However, based on the request for an application, we may need to show selective aggregation types on our own. This can be achieved using the [AggregateTypes](#) property.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Type(Syncfusion.EJ2.PivotView.SummaryTypes.DifferenceFrom).BaseField("Country").BaseItem("France").Add();
 values.Name("Amount").Caption("Sold Amount").Type(Syncfusion.EJ2.PivotView.SummaryTypes.Min).Add();
 }).AggregateTypes(new List<string>()
 { "DistinctCount", "Avg", "Product" }).ShowGroupingBar(true).Render()
```

### AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

| Drop filter here |          | FY 2015    |             | FY 2016    |
|------------------|----------|------------|-------------|------------|
| Country          | Products | Units Sold | Sold Amount | Units Sold |
| France           |          | 450        | \$714,955   | 526        |
| Germany          |          | 440        | \$563,515   | 496        |
| United States    |          | 546        | \$754,515   | 636        |
| Grand Total      |          | 1436       | \$2,032,985 | 1658       |

#### *Hiding aggregation type from button text*

By default, in value axis each field would be displayed by its name and aggregation type together. To hide aggregation type and display field name alone, set the property [ShowAggregationOnValueField](#) in [PivotViewDataSourceSettings](#) class to **false**.

#### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ShowAggregationOnValueField(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Type(Syncfusion.EJ2.PivotView.SummaryTypes.Sum).Add();
})).Render()
```

#### **AGGREGATION.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



|     |               |
|-----|---------------|
|     |               |
|     | Σ Values      |
| ↑ ≡ | Unit Sold ▼   |
| ↑ ≡ | Sold Amount ▼ |
|     |               |

&lt;br/&gt;

|                 |                  |
|-----------------|------------------|
| Units Sold ▼ ×  | Drop filter here |
| Sold Amount ▼ × | Year ↑ ≡ ×       |
| Country ↑ ≡ ×   | ► FY 2015        |

*Hiding aggregation type icon from UI*

By default, the icon to set aggregation type is enabled in the grouping bar. To disable this icon, set the property [ShowValueTypeIcon](#) in [PivotViewGroupingBarSettings](#) class to **false**.

**Note:** Icon to change the aggregation type can be hidden only in Grouping Bar but not in Field List at the moment.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).GroupingBarSettings(new
Syncfusion.EJ2.PivotView.PivotViewGroupingBarSettings { ShowValueTypeIcon =
false }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ShowGroupingB
ar(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold
Amount").Type(Syncfusion.EJ2.PivotView.SummaryTypes.Sum).Add();
})).Render()
```

**AGGREGATION.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
```

```
return View();
}
```

|                      |                  |
|----------------------|------------------|
| Sum of Units Sold ✕  | Drop filter here |
| Sum of Sold Amount ✕ | Year ↑ = ✕       |
| Country ↑ = ✕        | ► FY 2015        |

### Event

#### AggregateCellInfo

The event `AggregateCellInfo` triggers every time while rendering each value cell. This allows user to change the cell value and skip formatting if applied. It has following parameters:

- `fieldName` - It holds current cell's field name.
- `row` - It holds current cell's row value.
- `column` - It holds current cell's row value.
- `value` - It holds value of current cell.
- `cellSets` - It holds raw data for the aggregated value cell.
- `rowCellType` - It holds row cell type value.
- `columnCellType` - It holds column cell type value.
- `aggregateType` - It holds aggregate type of the cell.
- `skipFormatting` - boolean property, it allows to skip formatting if applied.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).GroupingBarSettings(new
Syncfusion.EJ2.PivotView.PivotViewGroupingBarSettings { ShowValueTypeIcon =
false }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ShowGroupingB
ar(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold
Amount").Type(Syncfusion.EJ2.PivotView.SummaryTypes.Sum).Add();
})).Render()
```

### AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
```

```

ViewBag.DataSource = data;
return View();
}

```

### ActionBegin

The event [actionBegin](#) triggers when clicking and selecting the aggregate type via the dropdown icon in the value field button, which is present in both grouping bar and field list UI. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. For example, while performing aggregation, the action name will be shown as **Aggregate field**.
- **fieldInfo**: It holds the selected value field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel**: It allows user to restrict the current action.

In the following example, action taken during aggregation type selection via dropdown icon can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = true
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Aggregate field') {
 args.cancel = true;
 }
 }
</script>

```

### ACTIONBEGIN-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionComplete

The event [actionComplete](#) triggers when a UI action, such as applying aggregation using the dropdown icon via the value field button, which is present in both the grouping bar and the field list UI, is completed. This allows user to identify the current UI action being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. For example, after completing the aggregation, the action name will be shown as **Field aggregated**.
- **fieldInfo**: It holds the selected value field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = true
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'Field aggregated') {
 // Triggers when the aggregation type is applied.
 }
 }
</script>
```

### ACTIONCOMPLETE-AGGREGATION.CS

```
public ActionResult Index()
{
```

```

var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```

### ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. For example, if the action fails while performing the aggregation, then the action name will be shown as **Aggregate field**.
- **errorInfo**: It holds the error information of the current UI action.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .ShowGroupingBar(true).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Aggregate field') {
 // Triggers when the current UI action fails to achieve the
desired result.
 }
 }
}
</script>

```

### ACTIONFAILURE-AGGREGATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Calculated Field in ASP.NET MVC Pivot Table Component

Allows end user to create a new calculated field in the pivot table, based on available fields from the bound data source or using simple formula with basic arithmetic operators. It can be added at runtime through the built-in dialog, invoked from Field List UI. To do so, set the [AllowCalculatedField](#) property in [PivotView](#) class to **true** in the pivot table. End user can now see a "CALCULATED FIELD" button enabled in Field List UI automatically, which on clicking will invoke the calculated field dialog and perform necessary operation.

Calculated field can also be included in the pivot table through code behind using the [PivotViewCalculatedFieldsSettings](#) class. The required properties to create a new calculate field are:

- [Name](#): It allows to indicate the calculated field with a unique name.
- [Formula](#): It allows to set the formula.
- [Format](#): It helps to set the number format for the resultant value.

**Note:** The calculated field is applicable only for value fields. Also, calculated field created through code behind will be automatically listed in the UI dialog as well.

### CSHTML

```
@{var amount = "\" + "Sum(Amount)" + "\"; }
@{var sold = "\" + "Sum(Sold)" + "\"; }
@{ var totalPrice = amount + "+" + sold; }
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
formatsettings.Name("Total").Format("C0").Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
values.Name("Total").Type(Syncfusion.EJ2.PivotView.SummaryTypes.CalculatedField).Add();
}).CalculatedFieldSettings(calculatedfieldsettings =>
{
calculatedfieldsettings.Name("Total").Formula(totalPrice).Add();
}).AllowCalculatedField(true).ShowFieldList(true).Render()
```

### CALCULATEDFIELD.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             |              | FY 2016    |         |
|---------------|------------|-------------|--------------|------------|---------|
|               | Units Sold | Sold Amount | Total Amount | Units Sold | Sold Am |
| France        | 450        | \$714,955   | \$715,405    | 526        | \$1,    |
| Germany       | 440        | \$563,515   | \$563,955    | 496        | \$1,    |
| United States | 546        | \$754,515   | \$755,061    | 636        | \$2,    |
| Grand Total   | 1436       | \$2,032,985 | \$2,034,421  | 1658       | \$5,    |

Meanwhile, user can also view calculated field dialog in UI by invoking `CreateCalculatedFieldDialog` method on an external button click which is shown in the below code sample.

### CSHTML

```
@{var amount = "\" + "Sum(Amount)" + "\"; }
@{var sold = "\" + "Sum(Sold)" + "\"; }
@{ var totalPrice = amount + "+" + sold; }
@Html.EJS().Button("calculated-field-btn").Content("Calculated
Field").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
formatsettings.Name("Total").Format("C0").Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
values.Name("Total").Type(Syncfusion.EJ2.PivotView.SummaryTypes.CalculatedField).Add();
}).CalculatedFieldSettings(calculatedfieldsettings =>
{
calculatedfieldsettings.Name("Total").Formula(totalPrice).Add();
```

```

 })).AllowCalculatedField(true).ShowFieldList(true).Render()
<script>
 document.getElementById("calculated-field-
btn").addEventListener('click', function () {
 var pivotObj =
document.getElementById("PivotView").ej2_instances[0];
 pivotObj.createCalculatedFieldDialog();
 });
</script>

```

## CALCULATEDFIELD.CS

```

public IActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Sold = 31, Amount = 52824, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 27, Amount = 46008, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 127800, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 69, Amount = 117576, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country =
"France", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 57, Amount = 85448, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 93, Amount = 139412, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
}

```



```

pivotData.Add(new PivotData { Sold = 35, Amount = 52470, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 28, Amount = 41977, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 48, Amount = 71957, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 36, Amount = 53969, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 25, Amount = 37480, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 69, Amount = 103436, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 16, Amount = 23989, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 28, Amount = 41977, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 19, Amount = 28486, Country =
"France", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 89, Amount = 141999.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 91, Amount = 145190.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 24, Amount = 38292, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 75, Amount = 119662.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 100, Amount = 159550, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 30, Amount = 47865, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 69, Amount = 110089.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 25, Amount = 39887.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 42, Amount = 67011, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 94, Amount = 149977, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 76, Amount = 121258, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 52, Amount = 82966, Country =
"France", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 33, Amount = 52651.5, Country =
"France", Products = "Touring Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 16, Amount = 23989, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 21, Amount = 33505.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 74, Amount = 126096, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 99, Amount = 148406, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 31, Amount = 49460.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 57, Amount = 97128, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });

```

```

pivotData.Add(new PivotData { Sold = 41, Amount = 61464, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 64, Amount = 102112, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 85, Amount = 144840, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 76, Amount = 129504, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 33, Amount = 56232, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 71, Amount = 120984, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 81, Amount = 138024, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 65, Amount = 110760, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 39, Amount = 66456, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 91, Amount = 155064, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 59, Amount = 100536, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 36, Amount = 61344, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 39, Amount = 58466, Country =
"Germany", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 47, Amount = 70458, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 19, Amount = 28486, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 34, Amount = 50971, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 34, Amount = 50971, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 26, Amount = 38979, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 15, Amount = 22490, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 79, Amount = 118426, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 14, Amount = 20991, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 23932.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 47, Amount = 74988.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 93, Amount = 148381.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 13, Amount = 20741.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 44, Amount = 70202, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q3" });

```

```

pivotData.Add(new PivotData { Sold = 59, Amount = 94134.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 34, Amount = 54247, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 48, Amount = 76584, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 35, Amount = 55842.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 71, Amount = 113280.5, Country =
"Germany", Products = "Touring Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 77, Amount = 131208, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 92, Amount = 156768, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q2" });
pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 91, Amount = 155064, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 56, Amount = 95424, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q2" });
pivotData.Add(new PivotData { Sold = 14, Amount = 23856, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 24, Amount = 40896, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 39, Amount = 66456, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q2" });
pivotData.Add(new PivotData { Sold = 84, Amount = 143136, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 40, Amount = 68160, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 96, Amount = 163584, Country =
"United Kingdom", Products = "Mountain Bikes", Year = "FY 2018", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 24, Amount = 35981, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"
});
pivotData.Add(new PivotData { Sold = 86, Amount = 128919, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"
});

```

```

 pivotData.Add(new PivotData { Sold = 31, Amount = 46474, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 36, Amount = 53969, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 40, Amount = 59965, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 69, Amount = 103436, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 95, Amount = 142410, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 95, Amount = 142410, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 30, Amount = 44975, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 11, Amount = 16494, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 97, Amount = 145408, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 16, Amount = 23989, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 40, Amount = 59965, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 68, Amount = 101937, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 11, Amount = 16494, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 27, Amount = 40478, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 45, Amount = 67460, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 100, Amount = 149905, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 70, Amount = 104935, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 100, Amount = 149905, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});

```

```

 pivotData.Add(new PivotData { Sold = 18, Amount = 26987, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 70, Amount = 104935, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 81, Amount = 121424, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country =
"United Kingdom", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 99, Amount = 148406, Country =
"United States", Products = "Road Bikes", Year = "FY 2018", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 43, Amount = 73272, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 43, Amount = 73272, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 52, Amount = 88608, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 91, Amount = 155064, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 37, Amount = 63048, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 41, Amount = 69864, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 23, Amount = 39192, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 85, Amount = 144840, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 28, Amount = 47712, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 53, Amount = 90312, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2018", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 82, Amount = 130831, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q1" });

```

```

 pivotData.Add(new PivotData { Sold = 41, Amount = 65415.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 60, Amount = 95730, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 71, Amount = 113280.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 45, Amount = 71797.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 21, Amount = 33505.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 94, Amount = 149977, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 34, Amount = 54247, Country =
"United States", Products = "Touring Bikes", Year = "FY 2015", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 14, Amount = 22337, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 76, Amount = 121258, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 50, Amount = 79775, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 119662.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 78179.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 40, Amount = 63820, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 94, Amount = 149977, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 17, Amount = 27123.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 45, Amount = 71797.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 56, Amount = 89348, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 119662.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 11, Amount = 17550.5, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q2" });

```

```

 pivotData.Add(new PivotData { Sold = 54, Amount = 86157, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 14, Amount = 22337, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 11, Amount = 17550.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 76, Amount = 121258, Country =
"United States", Products = "Touring Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 45, Amount = 71797.5, Country =
"United Kingdom", Products = "Touring Bikes", Year = "FY 2018", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 80, Amount = 127640, Country =
"United States", Products = "Touring Bikes", Year = "FY 2018", Quarter =
"Q1" });
 return pivotData;
 }
}
public class PivotData
{
 public int Sold { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string Products { get; set; }
 public string Year { get; set; }
 public string Quarter { get; set; }
}

```

| CALCULATED FIELD |            |             |            |             |            |
|------------------|------------|-------------|------------|-------------|------------|
|                  | FY 2015    |             | FY 2016    |             | FY 2017    |
|                  | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France           | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany          | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States    | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total      | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |



### Create Calculated Field

×

Enter the field name

Drag and drop fields to formula

Country (Count)

Products (Count)

Quarter (Count)

Formula

Example: ("Sum(Order\_Count)" + "Sum(In\_Stock)") \* 250

Format

None

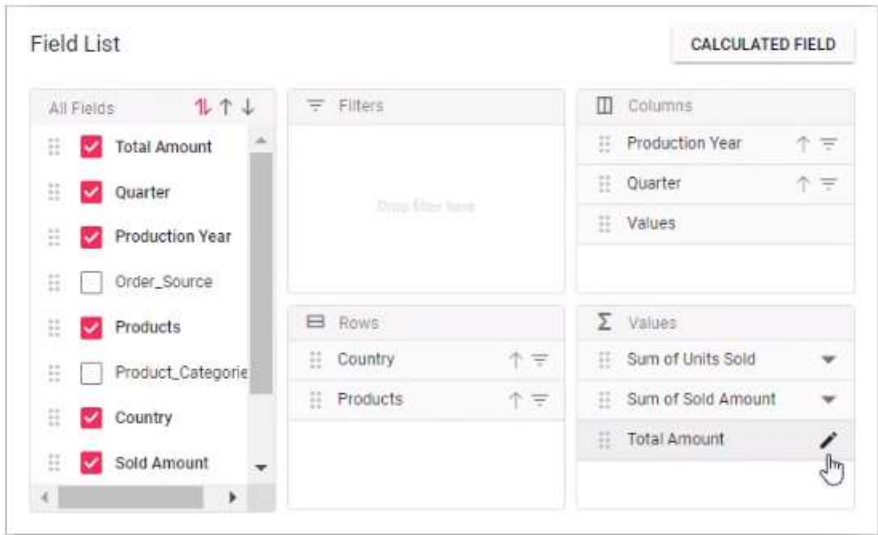
Enter custom format string

OK CANCEL

#### *Editing through the field list and the grouping bar*

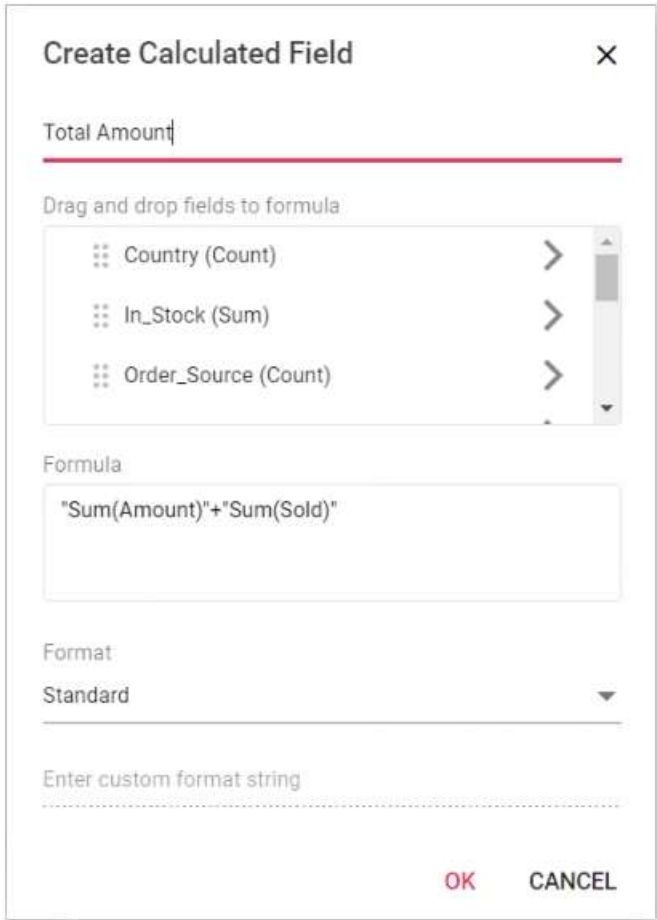
User can also modify the existing calculated field using the built-in edit option available directly in the field list (or) grouping bar. To do so, click the "Edit" icon available in the calculated field button. Now the calculated field dialog is opened and the current calculated field name, formula and format can be changed at runtime.





<br/>

<br/>



*Renaming the existing calculated field*

Existing calculated field can be renamed only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing name getting

displayed in the text box at the top of the dialog. Now, change the name based on user requirement and click "OK".

<!-- markdownlint-disable MD012 -->

**Create Calculated Field** [X]

Enter the field name

Drag and drop fields to formula

- Sold Amount (Sum)
- Total Amount (Calculated Field)** [Edit]
- Units Sold (Sum)

Formula

Example: ("Sum(Order\_Count)" + "Sum(In\_Stock)") \* 250

Format

None

Enter custom format string

OK CANCEL

<br/>

<br/>

**Create Calculated Field** [X]

Total Sales Amount

Drag and drop fields to formula

- Sold Amount (Sum) >
- Total Amount (Calculated Field)** [Edit] [Delete]
- Units Sold (Sum) >

Formula

"Sum(Amount)"+"Sum(Sold)"

Format

Standard

Enter custom format string

OK CANCEL

#### *Editing the existing calculated field formula*

Existing calculated field formula can be edited only through the UI at runtime. To do so, open the calculated field dialog, select the target field and click "Edit" icon. User can now see the existing formula getting displayed in a multiline text box at the bottom of the dialog. Now, change the formula based on user requirement and click "OK".



### Create Calculated Field

×

Enter the field name

Drag and drop fields to formula

Sold Amount (Sum) >

Total Amount (Calculated Field)  

Units Sold (Sum) >

Formula

Example: ("Sum(Order\_Count)" + "Sum(In\_Stock)") \* 250

Format

None

Enter custom format string

OK CANCEL

&lt;br/&gt;

&lt;br/&gt;

Create Calculated Field

×

Total Amount

Drag and drop fields to formula

Sold Amount (Sum)

>

Total Amount (Calculated Field)

✕

👉

Units Sold (Sum)

>

Formula

"Sum(Amount)+"Sum(Sold)"

Format

Standard

Enter custom format string

OK

CANCEL

#### *Reusing the existing formula in a new calculate field*

While creating a new calculated field, if user wants to add the formula of an existing calculated field, it can be done easily. To do so, simply drag-and-drop the existing calculated field to the "Formula" section.

Create Calculated Field

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum)

Total Amount (Calculated Field)

Total Amount (Calculated Field)

Formula

Example: ("Sum(Order\_Count)" + "Sum(In\_Stock)") \* 250

Format

None

Enter custom format string

OK

CANCEL

<br/>

<br/>

Copyright © 2001 -2024 Syncfusion Inc.

1450

Create Calculated Field

×

Sales Amount

Drag and drop fields to formula

Sold Amount (Sum)

>

Total Amount (Calculated Field)

✕

✎

Units Sold (Sum)

>

Formula

Example: ("Sum(Order\_Count)" + "Sum(In\_Stock)") \* 250

Total Amount (Calculated Field)

Format

None

Enter custom format string

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

**Create Calculated Field** [X]

Sales Amount

Drag and drop fields to formula

- Sold Amount (Sum) >
- Total Amount (Calculated Field) [trash] [pencil]
- Units Sold (Sum) >

Formula

"Sum(Amount)"+"Sum(Sold)"

Format

None

Enter custom format string

OK CANCEL

#### *Apply the format to the calculated field values*

Values in a new or existing calculated field can be formatted via the calculated field UI or code behind. The [FormatSettings](#) property in code-behind can be used to specify the desired format. For more information about the supported formats refer [refer here](#).

To apply format to calculated field values at runtime via UI, a built-in dropdown under the "Format" label is available, from which the user can select the pre-defined format options listed below.

- **Standard** - Denotes the numeric type.
- **Currency** - Denotes the currency type.
- **Percent** - Denotes the percentage type.
- **Custom** - Denotes the custom format. For example: "C2". This shows the value "9584.3" as "\$9584.30."
- **None** - Denotes that no format will be applied.

**Note:** By default, **None** will be selected from the dropdown.



The screenshot shows the 'Create Calculated Field' dialog box. At the top, the title is 'Create Calculated Field' with a close button (X). Below the title, the text 'Total Amount' is displayed. A section labeled 'Drag and drop fields to formula' contains a list of fields: 'Sold Amount (Sum)', 'Total Amount (Calculated Field)' (highlighted in red), and 'Units Sold (Sum)'. Below this, the 'Formula' section shows the formula: `"Sum(Amount)"+"Sum(Sold)"`. The 'Format' section is expanded, showing a dropdown menu with options: 'Standard' (selected and highlighted in red), 'Currency', 'Percent', 'Custom', and 'None'.

In addition, you can specify the desired custom formats by selecting the **Custom** option from the "Format" dropdown.

*Supported operators and functions for the calculated field formula*

Below is a list of operators and functions that can be used in the formula to create the calculated fields.

- **+** – addition operator.

`typescript

Syntax:  $X + Y$

,

- **-** – subtraction operator.

`typescript

Syntax:  $X - Y$

,

- **\*** – multiplication operator.

`typescript

Syntax:  $X * Y$

- `/` – division operator.

`typescript`

Syntax: X / Y

- `^` – power operator.

### Grouping in ASP.NET MVC Pivot Table Component

**Note:** This feature is applicable only for relational data source.

Grouping is the most-useful feature in pivot table and the component automatically groups date, time, number and string. For example, the date type can be formatted and displayed based on year, quarter, month, and more. Likewise, the number type can be grouped range-wise, such as 1-5, 6-10, etc. These group fields will act as individual fields and allows users to drag them between different axes such as columns, rows, values, and filters and create pivot table at runtime.

The grouping can be enabled by setting the [AllowGrouping](#) property in [PivotView](#) class to **true**.

To perform the grouping action via UI, right click on the pivot table's row or column header, select "**Group**", a dialog will appear in which fill the appropriate options to group the data. To ungroup, right click on the pivot table's row or column header, select "**Ungroup**".

The following are the three different types of grouping:

- Number Grouping
- Date Grouping
- Custom Grouping

**Note:** Similar to Excel, only one type of grouping can be applied for a field.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
 formatsettings.Name("Product_ID").Format("N0").Add();
 formatsettings.Name("Date").Type("date").Format("dd/MM/yyyy-hh:mm
a").Add();
 })
 .Rows(rows => { rows.Name("Date").Add(); })
 .Columns(columns => { columns.Name("Product_ID").Caption("Product
ID").Add(); })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 })
 })
```

```
values.Name("Amount").Caption("Sold Amount").Add(); })).Render()
```

### **GROUPING.CS**

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```

#### *Number Grouping*

Number grouping allows users to organize data, which is in number format into different ranges, such as 1-5, 6-10, etc. Number grouping can be configured via UI, by right-clicking on the number based header in the pivot table.

### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
 formatsettings.Name("Product_ID").Format("N0").Add();
 })
 .Rows(rows => { rows.Name("Product_ID").Caption("Product ID").Add(); })
 .Columns(columns => { columns.Name("Products").Add(); })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add(); })).Render()
```

### **GROUPING.CS**

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```

| Product ID | Bottles and Cages |             |
|------------|-------------------|-------------|
|            | Units Sold        | Sold Amount |
| 1001       | 162               | \$344.00    |
| 1002       |                   |             |
| 1003       |                   |             |
| 1004       |                   |             |
| 1005       |                   |             |
| 1006       |                   |             |

#### Range selection

The "**Starting at**" and "**Ending at**" options are used to set the number range depending on which the headers will be grouped. For example, if the "Product\_ID" field holds the number from "1001" to "1010" and the user chooses to group the number range by setting "**1004**" to "**Starting at**" and "**1008**" to "**Ending at**" options on their own. Then the specified number range will be used for number grouping and the rest will be grouped as "**Out of Range**".

Grouping

☒ Starting at

1004

☒ Ending at

1008

Interval by

Enter value

OK

CANCEL

#### Range interval

The "**Interval by**" option is used to separate the selected number data type field into range-wise such as 1-5, 6-10, etc.

For example, if the user wants to display the "ProductID" data field with a group interval of "2" by setting the "Interval by" option on their own. The "ProductID" field will then be grouped by the specified range of intervals, such as "**1004-1005**", "**1006-1007**", etc.

Grouping

☒ Starting at

1004

▼ ▲

☒ Ending at

1008

▼ ▲

Interval by

2

X ▼ ▲

OK

CANCEL

&lt;br/&gt;

|                        |                   |             |           |             |     |
|------------------------|-------------------|-------------|-----------|-------------|-----|
| Sum of Unit Sold ▼ ✕   | Drop filter here  |             |           |             |     |
| Sum of Sold Amount ▼ ✕ | Products ↑ ≡ ✕    |             |           |             |     |
| Product ID ↑ ≡ ✕       | Bottles and Cages |             | Cleaners  |             | Fen |
|                        | Unit Sold         | Sold Amount | Unit Sold | Sold Amount | Uni |
| 1004-1005              |                   |             |           |             |     |
| 1006-1007              |                   |             |           |             |     |
| 1008                   |                   |             |           |             |     |
| Out of Range           | 162               | \$344.00    | 297       | \$939.00    |     |
| Grand Total            | 162               | \$344.00    | 297       | \$939.00    |     |

Number grouping can also be configured using the [PivotViewGroupSettings](#) class through code-behind. The properties required are:

- [Name](#): Allows user to set the field name.
- [RangeInterval](#): Allows user to set the interval between two numbers.
- [StartingAt](#): Allows user to set the starting number.
- [EndingAt](#): Allows user to set the ending number.
- [Type](#): Allows user to set the group type. For number grouping, [Number](#) is set.

**Note:** If starting and ending numbers specified in [StartingAt](#) and [EndingAt](#) properties are in-between the number range, then rest of the numbers will be grouped and placed in “Out of Range” section introduced specific to this feature.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
 formatsettings.Name("Product_ID").Format("N0").Add();
 })
 .Rows(rows => { rows.Name("Product_ID").Caption("Product ID").Add(); })
 .Columns(columns => { columns.Name("Products").Add(); })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 .GroupSettings(groupsettings => {
groupsettings.Name("Product_ID").Type("Number").RangeInterval(2).StartingAt(
1004).EndingAt(1008).Add();
 })).Render()
```

### GROUPING.CS

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```

|                        |                   |             |           |             |     |
|------------------------|-------------------|-------------|-----------|-------------|-----|
| Sum of Unit Sold ▾ ⊗   | Drop filter here  |             |           |             |     |
| Sum of Sold Amount ▾ ⊗ | Products ↑ ≡ ⊗    |             |           |             |     |
| Product ID ↑ ≡ ⊗       | Bottles and Cages |             | Cleaners  |             | Fen |
|                        | Unit Sold         | Sold Amount | Unit Sold | Sold Amount | Uni |
| 1004-1005              |                   |             |           |             |     |
| 1006-1007              |                   |             |           |             |     |
| 1008                   |                   |             |           |             |     |
| Out of Range           | 162               | \$344.00    | 297       | \$939.00    |     |
| Grand Total            | 162               | \$344.00    | 297       | \$939.00    |     |

### Ungrouping the existing number groups

By right-clicking the appropriate header and selecting **"Ungroup"** from the context menu in the pivot table component, users can ungroup the applied number grouping.

| Product ID <span>↑ ≡ ×</span> | Bottles and Cages |             |
|-------------------------------|-------------------|-------------|
|                               | Unit Sold         | Sold Amount |
| 1004-1005                     |                   |             |
| 1006-1007                     |                   |             |
| 1008                          |                   |             |
| Out of Range                  | 162               | \$344.00    |
| Grand Total                   | 162               | \$344.00    |

### Date Grouping

Date grouping allows users to organize data, which is in date format into different sections such as years, quarters, months, days, hours, minutes, and seconds. Date grouping can be configured via UI, by right-clicking on the date and time based header in the pivot table.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
formatsettings.Name("Date").Type("date").Format("dd/MM/yyyy-hh:mm a").Add();
}))
.Rows(rows => { rows.Name("Date").Add(); })
.Columns(columns => {
columns.Name("Product_Categories").Caption("Product Categories").Add(); })
.Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add(); })}).Render()
```

### GROUPING.CS

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```



| Date                | 1001      |             |
|---------------------|-----------|-------------|
|                     | Unit Sold | Sold Amount |
| 01/01/2015-08:18 PM | 6         | \$12.00     |
| 05/01/2015-08:19 PM |           |             |
| 02/02/2015-10:22 AM |           |             |
| 10/02/2015-10:23 AM |           |             |
| 20/02/2015-11:25 AM |           |             |
| 07/03/2015-05:11 AM | 12        | \$24.00     |

#### Range selection

The "**Starting at**" and "**Ending at**" options are used to set the date range depending on which the headers will be grouped. For example, if the "Date" field holds the date from "01/01/2015" to "02/12/2018" and the user chooses to group the date range by setting "**01/07/2015**" to "**Starting at**" and "**31/07/2017**" to "**Ending at**" options on their own. Then the specified date range will be used for date grouping and the rest will be considered as "**Out of Range**".

Grouping

☒ Starting at

01/07/2015 12:00:00 AM

☒ Ending at

31/07/2017 12:00:00 AM

Interval by

Select groups

OK

CANCEL

#### Group interval

The "**Interval by**" option is used to separate the selected date fields into years, quarters, months, days, hours, minutes and seconds. For example, if the user wants to display the "Date" field with group intervals as "**Years**" and "**Months**" by selecting the "**Interval by**" option on their own. The "Date" field will then be separated by the specified group intervals and created as two new fields, namely "**Years**"

**(Date)**" which holds the date years and **"Months (Date)"** which holds the date months. Such fields can be used for report manipulations in the pivot table at runtime.

**Note:** When none of the **Interval by** options are chosen, the **OK** button in the dialog will be disabled, meaning that at least one interval option should be selected in order to apply the date grouping.

Grouping

☒ Starting at

01/07/2015 12:00:00 AM

☒ Ending at

31/07/2017 12:00:00 AM

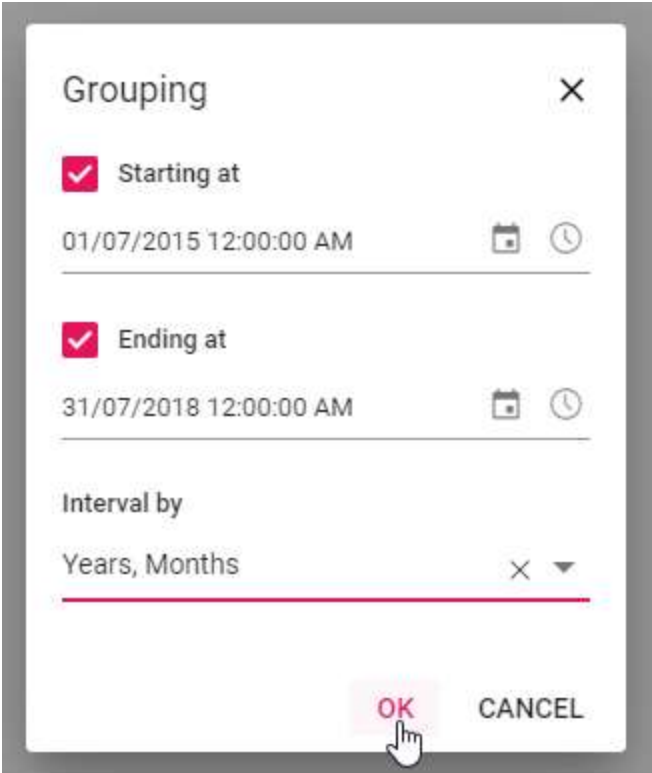
Interval by

Years, Months

e.g: Months

- ☐ Minutes
- ☐ Hours
- ☐ Days
- ☒ Months
- ☐ Quarter Year
- ☐ Quarters
- ☒ Years

<br/>



<br/>

|                    |                    |                  |
|--------------------|--------------------|------------------|
| Sum of Unit Sold   | Products (All)     | Product ID (All) |
| Sum of Sold Amount | Product Categories |                  |
| Years (Date)       | Accessories        | Bikes            |
| Months (Date)      | Unit Sold          | Sold Amount      |
| 2015               | 18                 | \$60.00          |
| Sep                | 15                 | \$36.00          |
| Oct                | 3                  | \$24.00          |
| 2016               | 180                | \$668.00         |
| 2017               | 60                 | \$248.00         |
| Out of Range       | 327                | \$1,363.00       |
| Out of Range       | 327                | \$1,363.00       |
| Grand Total        | 585                | \$2,339.00       |

Date grouping can also be configured using the [PivotViewGroupSettings](#) class through code-behind. The properties required are:

- [Name](#): Allows user to set the field name.
- [Type](#): Allows user to set the group type. For date grouping, [Date](#) is set.

- [StartingAt](#): Allows user to set starting date.
- [EndingAt](#): Allows user to set ending date.
- [GroupInterval](#): Allows user to set interval in year, quarter, month, day, hour, minute, or second pattern.

**Note:** From the date format "YYYY-DD-MM HH:MM:SS", if user wants to display only year and month, then the [GroupInterval](#) property should be set with **Years** and **Months** alone. Also, user can shuffle the order of year, quarter, month, day, hour, minute, or second based on their requirement and display the same in the pivot table.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
 formatsettings.Name("Date").Type("date").Format("dd/MM/yyyy-hh:mm
a").Add();
 })
 .Rows(rows => { rows.Name("Date").Add(); })
 .Columns(columns => {
columns.Name("Product_Categories").Caption("Product Categories").Add(); })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add(); })
 .Filters(filters =>
 {
 filters.Name("Products").Add();
 filters.Name("Product_ID").Caption("Product ID").Add(); })
 .GroupSettings(groupsettings => {
groupsettings.Name("Date").Type("Date").GroupInterval("@(new List<string>()
{ "Years", "Months" })").StartingAt("2015-07-01").EndingAt("2017-07-
31").Add();
 })).Render()
```

### GROUPING.CS

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```

Furthermore, in the field list UI, these date group fields **Years (Date)**, **Quarters (Date)**, **Months (Date)**, etc... will be automatically grouped and displayed under the **Date** folder name.

Field List

All Fields

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

▼ Date

☐

☑

Years (Date)

☐

☑

Quarters (Date)

☐

☑

Sold Amount

☐

☑

Products

☐

☑

Unit Sold

☐

☑

Product ID

☐

☑

Product Categories

☰ Filters

☐

Products (All)

☐

Product ID (All)

☰ Rows

☐

Years (Date)

☐

Quarters (Date)

☰ Columns

☐

Product Categories

☰ Values

☐

Sum of Unit Sold

☐

Sum of Sold Amount

CLOSE

Ungrouping the existing date groups

By right-clicking the appropriate header and selecting **"Ungroup"** from the context menu in the pivot table component, users can ungroup the applied date grouping.

| Years (Date) ↑ = ×  | 1010      |             |
|---------------------|-----------|-------------|
| Months (Date) ↑ = × | Unit Sold | Sold Amount |
| ▼ 2015              | 6         | \$12.00     |
| Sep                 | 6         | \$12.00     |
| Oct                 |           |             |
| ▼ 2016              | 40        | \$80.00     |
| Jan                 |           |             |
| Feb                 | 12        | \$24.00     |

Custom Grouping

Custom grouping can group any data type, such as date, time, number and string, into a custom field based on the user's needs. It can be configured via the UI by right-clicking on any header in the pivot table.

CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)).EnableSorting(true)
 .FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
 formatsettings.Name("Product_ID").Format("N0").Add();
 })
 .Rows(rows => { rows.Name("Products").Add(); })
 .Columns(columns => { columns.Name("Product_ID").Caption("Product
ID").Add(); })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add(); })
).Render()
```

### GROUPING.CS

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```

In order to create custom grouping in the pivot table, a minimum of two headers belonging to a specific field should be chosen. To select more than one header, press and hold the CTRL key or hold the SHIFT key and click the appropriate row or column headers. Once selection is done, right-click and select "Group".

| Products <span>↑</span> <span>≡</span> <span>×</span> | 1001      |             |
|-------------------------------------------------------|-----------|-------------|
|                                                       | Unit Sold | Sold Amount |
| Bottles and Cages                                     | 162       | \$344.00    |
| Cleaners                                              |           |             |
| Fenders                                               |           |             |
| Gloves                                                |           |             |
| Jerseys                                               |           |             |
| Mountain Bikes                                        |           |             |
| Road Bikes                                            |           |             |
| Shorts                                                |           |             |
| Touring Bikes                                         |           |             |

In the dialog, the "**Field caption**" is the alias name of the new custom field that will be used to shown up in the pivot table component.

Grouping

Field caption

Product category

Group name

Enter the caption to display in header

OKCANCEL

The "**Group Name**" option helps to set the name of the header to hold the other selected headers. For example, if the user wants to group headers such as "**Gloves**", "**Jerseys**" and "**Shorts**" in the "Products" field by setting the top level name as "**Clothings**" to "**Group Name**" on their own. The selected headers are then grouped under the name "**Clothings**" in the pivot table.

Grouping

Field caption

Product category

Group name

Clothings

OKCANCEL

<br/>

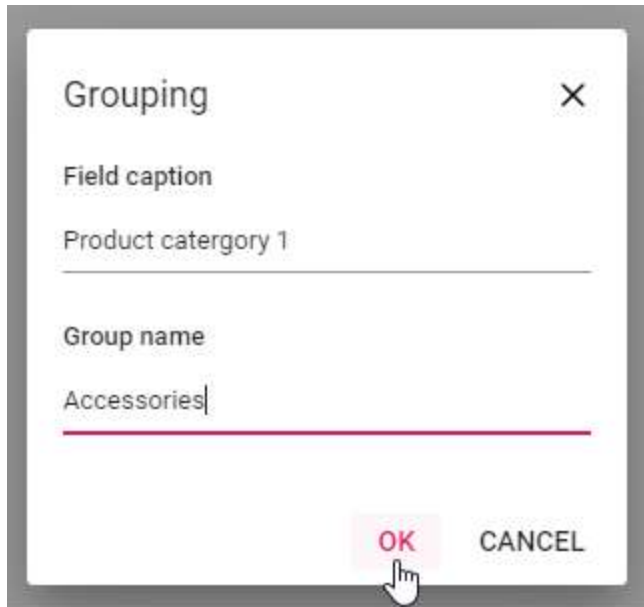
|                    |                  |             |           |             |           |
|--------------------|------------------|-------------|-----------|-------------|-----------|
| Sum of Unit Sold   | Drop filter here |             |           |             |           |
| Sum of Sold Amount | Product ID       |             |           |             |           |
| Product category   | 1001             |             | 1002      |             | 1003      |
| Products           | Unit Sold        | Sold Amount | Unit Sold | Sold Amount | Unit Sold |
| Bottles and Cages  | 162              | \$344.00    |           |             |           |
| Bottles and Cages  | 162              | \$344.00    |           |             |           |
| Cleaners           |                  |             | 297       | \$939.00    |           |
| Cleaners           |                  |             | 297       | \$939.00    |           |
| Clothings          |                  |             |           |             |           |
| Gloves             |                  |             |           |             |           |
| Jerseys            |                  |             |           |             |           |
| Shorts             |                  |             |           |             |           |
| Fenders            |                  |             |           |             |           |

User can also apply new custom grouping options to an existing custom field by right-clicking on the custom group header in the pivot table. For example, if the user wants to create a new custom group for the current custom group headers such as "Bottles and Cages", "Cleaners" and "Fenders" by setting the top level name as "Accessories" to "Group Name" on their own. The selected headers will then be grouped in the pivot table under the name "Accessories" with a new custom field called "Product category 1".

|                   |           |             |
|-------------------|-----------|-------------|
| Product category  | 1001      |             |
| Products          | Unit Sold | Sold Amount |
| Bottles and Cages | 162       | \$344.00    |
| Bottles and Cages | 162       | \$344.00    |
| Cleaners          |           |             |
| Cleaners          |           |             |
| Clothings         |           |             |
| Gloves            |           |             |
| Jerseys           |           |             |
| Shorts            |           |             |
| Fenders           |           |             |

<br/>





<br/>

|                           |                  |             |           |             |           |
|---------------------------|------------------|-------------|-----------|-------------|-----------|
| Sum of Unit Sold ▼ ✕      | Drop filter here |             |           |             |           |
| Sum of Sold Amount ▼ ✕    | Product ID ↑ ≡ ✕ |             |           |             |           |
| Product category... ↑ ≡ ✕ | 1001             |             | 1002      |             | 1003      |
| Product category... ↑ ≡ ✕ |                  |             |           |             |           |
| Products ↑ ≡ ✕            | Unit Sold        | Sold Amount | Unit Sold | Sold Amount | Unit Sold |
| ▼ Accessories             | 162              | \$344.00    | 297       | \$939.00    |           |
| ▼ Bottles and Cages       | 162              | \$344.00    |           |             |           |
| Bottles and Cages         | 162              | \$344.00    |           |             |           |
| ▼ Cleaners                |                  |             | 297       | \$939.00    |           |
| Cleaners                  |                  |             | 297       | \$939.00    |           |
| ▼ Fenders                 |                  |             |           |             |           |
| Fenders                   |                  |             |           |             |           |
| ▼ Clothings               |                  |             |           |             |           |

Custom grouping can also be configured using the [PivotViewGroupSettings](#) class through code-behind. The properties required are:

- [Name](#): Allows user to set the field name.
- [Caption](#): Allows user to set the caption name for custom grouping field.
- [CustomGroups](#): Allows user to set the custom groups.
- [Type](#): Allows user to set the group type. For custom grouping, [Custom](#) is set.

The available custom group properties in [CustomGroups](#) property are:

- [GroupName](#): Allows user to set the group name (or title) for selected headers.
- [Items](#): It allows to set the headers which needs to be grouped from display.

**Note:** When the [GroupName](#) with the headers listed in [Items](#) in the [CustomGroups](#) class is grouped by the defined [GroupName](#) and the rest is grouped by its own name in the pivot table.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("350").ShowGroupingBar(true).AllowGrouping(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C").Add();
 formatsettings.Name("Product_ID").Format("N0").Add();
 })
 .Rows(rows => { rows.Name("Products").Add(); })
 .Columns(columns => { columns.Name("Product_ID").Caption("Product
ID").Add(); })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add(); })
 .GroupSettings(groupsettings => {
groupsettings.Name("Products").Type("Custom").Caption("Product
category").CustomGroups(customgroup => {
customgroup.GroupName("Clothings").Items("@(new string[] { "Gloves",
"Jerseys", "Shorts" })").Add(); }).Add();
 })).Render()
```

### GROUPING.CS

```
public ActionResult Index()
{
 var data = GetGroupData();
 ViewBag.DataSource = data;
 return View();
}
```

|                        |                  |             |           |             |           |
|------------------------|------------------|-------------|-----------|-------------|-----------|
| Sum of Unit Sold ▼ ✕   | Drop filter here |             |           |             |           |
| Sum of Sold Amount ▼ ✕ | Product ID ↑ ≡ ✕ |             |           |             |           |
| Product categ... ↑ ≡ ✕ | 1001             |             | 1002      |             | 1003      |
| Products ↑ ≡ ✕         | Unit Sold        | Sold Amount | Unit Sold | Sold Amount | Unit Sold |
| ▼ Bottles and Cages    | 162              | \$344.00    |           |             |           |
| Bottles and Cages      | 162              | \$344.00    |           |             |           |
| ▼ Cleaners             |                  |             | 297       | \$939.00    |           |
| Cleaners               |                  |             | 297       | \$939.00    |           |
| ▼ Clothings            |                  |             |           |             |           |
| Gloves                 |                  |             |           |             |           |
| Jerseys                |                  |             |           |             |           |
| Shorts                 |                  |             |           |             |           |
| ▼ Fenders              |                  |             |           |             |           |

### Ungrouping the existing custom groups

By right-clicking the appropriate header and selecting "**Ungroup**" from the context menu in the pivot table component, users can ungroup the applied custom grouping.

**Note:** When a specific field is removed from the report after ungrouping, its custom group fields will also be removed from the pivot table.

|                        |           |             |
|------------------------|-----------|-------------|
| Product categ... ↑ ≡ ✕ | 1001      |             |
| Products ↑ ≡ ✕         | Unit Sold | Sold Amount |
| ▼ Bottles and Cages    | 162       | \$344.00    |
| Bottles and Cages      | 162       | \$344.00    |
| ▼ Cleaners             |           |             |
| Cleaners               |           |             |
| ▼ Clothings            |           |             |
| Gloves                 |           |             |
| Jerseys                |           |             |
| Shorts                 |           |             |

Group

Ungroup

### Filtering

Filtering allows to view the pivot table with selective records based on members that can be either included or excluded through UI and code-behind.

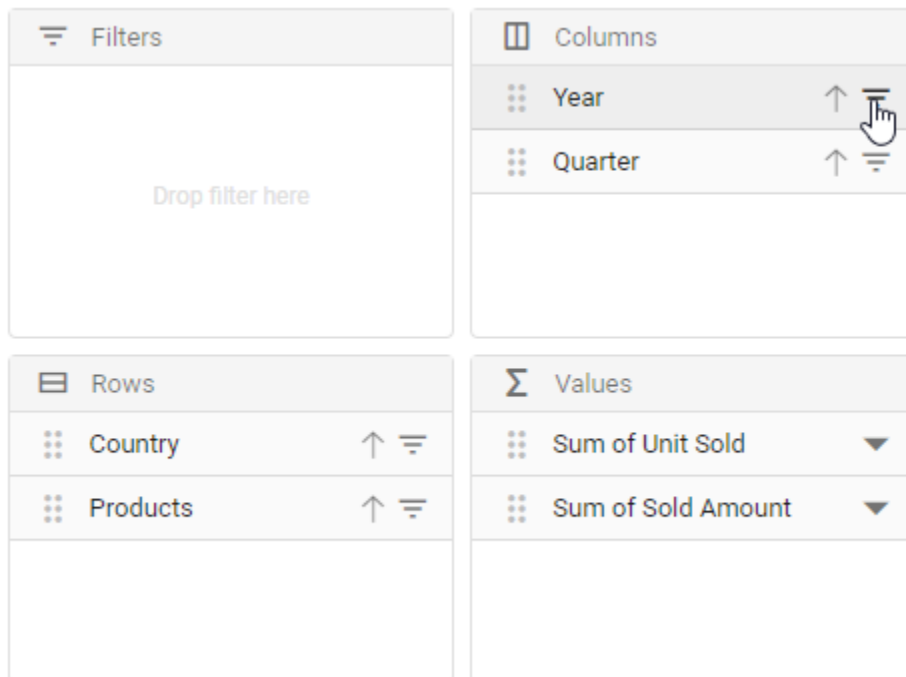
The following are the three different types of filtering:

- Member filtering
- Label filtering
- Value filtering

**Note:** When all the above filtering options are disabled via code-behind, then the filter icon would be disabled in the field list or grouping bar UI.

#### Member filtering

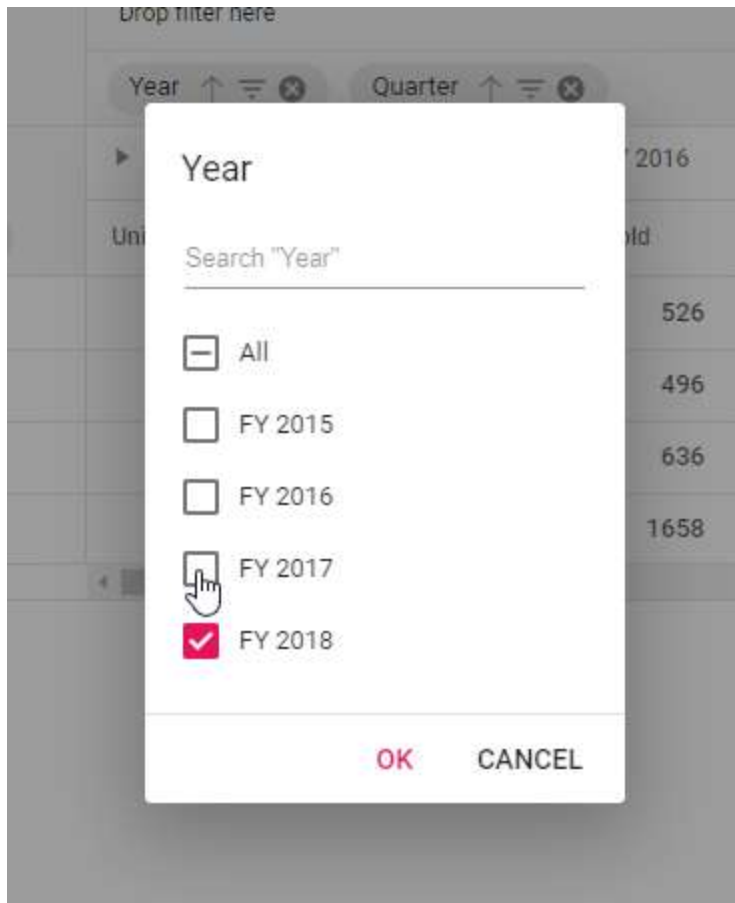
Allows to view the pivot table with selective records based on included and excluded members in each field. By default, member filter option is enabled by the [AllowMemberFilter](#) boolean property in [PivotViewDataSourceSettings](#) class. This UI option helps end user to filter members by clicking the filter icon besides any field in the row, column and filter axes available in the field list or grouping bar UI at runtime.



<br/>

|                    |                  |             |           |             |           |  |
|--------------------|------------------|-------------|-----------|-------------|-----------|--|
| Sum of Unit Sold   | Drop filter here |             |           |             |           |  |
| Sum of Sold Amount | Year             |             | Quarter   |             |           |  |
| Country            | FY 2015          |             | FY 2016   |             |           |  |
| Products           | Unit Sold        | Sold Amount | Unit Sold | Sold Amount | Unit Sold |  |
| France             | 450              | \$714,955   | 526       | \$1,542,104 |           |  |
| Germany            | 440              | \$563,515   | 496       | \$1,772,104 |           |  |
| United States      | 546              | \$754,515   | 636       | \$2,263,104 |           |  |
| Grand Total        | 1436             | \$2,032,985 | 1658      | \$5,577,312 |           |  |

<br/>



<br/>

| Sum of Unit Sold   |  | Drop filter here         |             |                       |
|--------------------|--|--------------------------|-------------|-----------------------|
| Sum of Sold Amount |  | Year ↑ ⇅ × Quarter ↑ ⇅ × |             |                       |
| Country ↑ ⇅ ×      |  | FY 2018                  |             | Grand Total           |
| Products ↑ ⇅ ×     |  | Unit Sold                | Sold Amount | Unit Sold Sold Amount |
| ▶ France           |  | 16                       | \$27,264    | 16 \$27,264           |
| ▶ Germany          |  | 96                       | \$77,264    | 96 \$77,264           |
| ▶ United States    |  | 76                       | \$97,264    | 76 \$97,264           |
| Grand Total        |  | 188                      | \$201,792   | 188 \$201,792         |

Meanwhile filtering can also be configured at code behind using the [PivotViewFilterSettings](#) class while initial rendering of the component. The basic settings required to add filter criteria are:

- [Name](#): It allows to set the appropriate field name.
- [Type](#): It allows to set the filter type as [FilterType.Include](#) or [FilterType.Exclude](#) to include or exclude field members respectively.
- [Items](#): It allows to set the members which needs to be either included or excluded from display.

- [LevelCount](#): It allows to set level count of the field to fetch data from the cube. **This property applicable only for OLAP data source.**

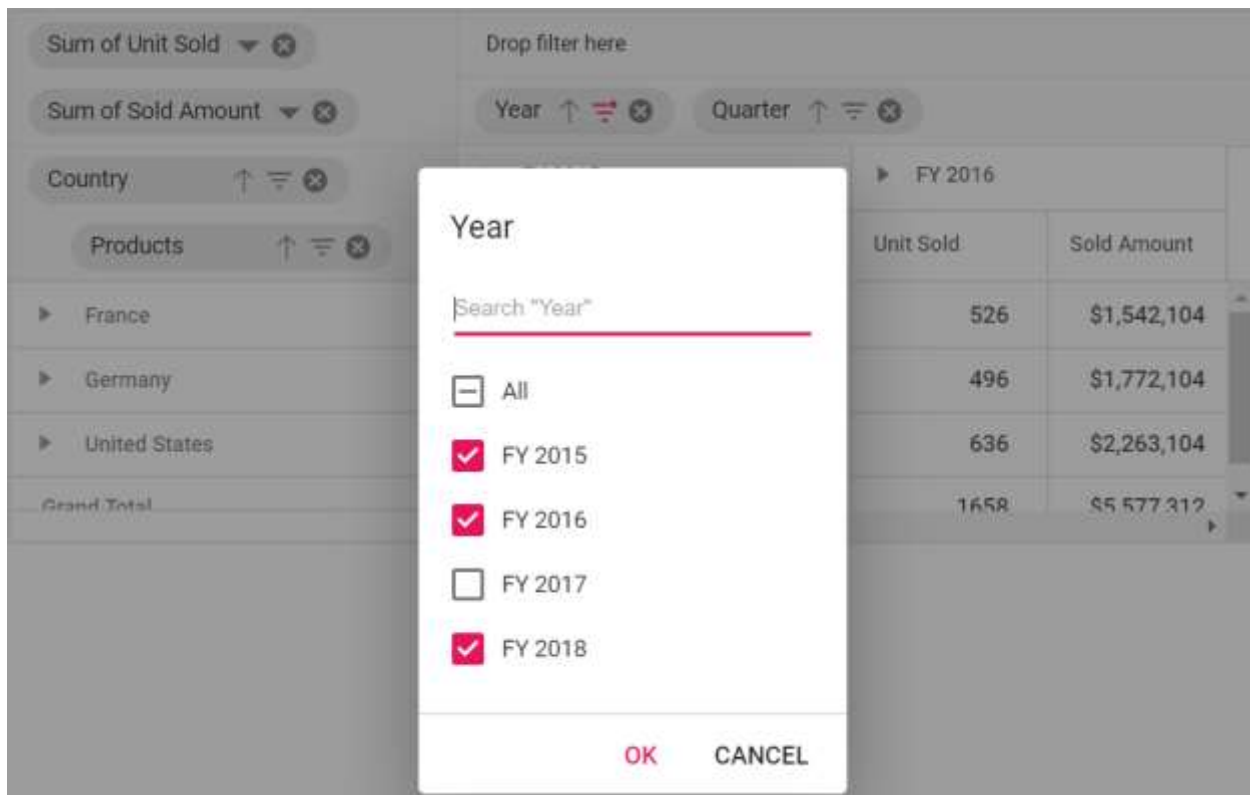
**Note:** When specifying unavailable or inappropriate members to include or exclude filter items collection, they will be ignored.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FilterSettings(filterSettings =>
{
filterSettings.Name("Year").Type(Syncfusion.EJ2.PivotView.FilterType.Exclude)
.Items(ViewBag.filterMembers).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

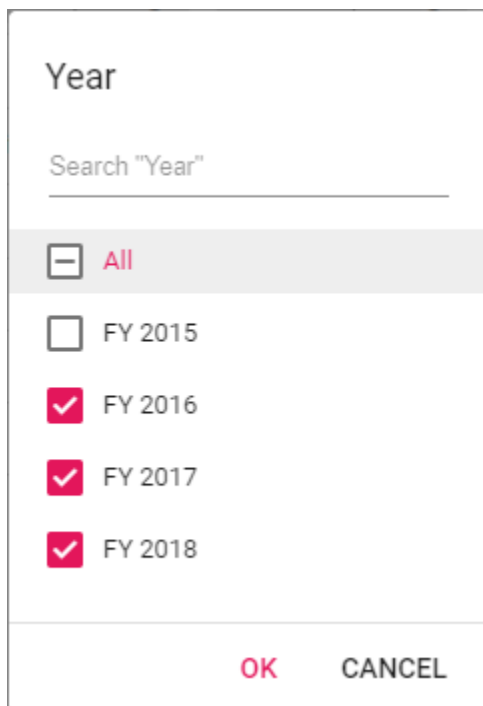
### FILTERING.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
ViewBag.filterMembers = new string[] { "FY 2017" };
return View();
}
```



#### [Option to select and unselect all members](#)

The member filter dialog comes with an option "All", which on checked selects all members and on unchecked deselects all members. The option "All" would appear in intermediate state mentioning that both selected and unselected child members are available.



When all members are deselected, the "Ok" button in member filter dialog would be disabled, meaning, at least one member should be selected and bound to the pivot table component.

A member filter dialog box titled "Year". It features a search bar with the placeholder text "Search 'Year'". Below the search bar, there is a list of members: "All", "FY 2015", "FY 2016", "FY 2017", and "FY 2018". Each member has an unchecked checkbox to its left. The "All" member is highlighted with a light gray background. At the bottom of the dialog, there are two buttons: "OK" and "CANCEL". The "OK" button is disabled (grayed out).

[Provision to search specific member\(s\)](#)

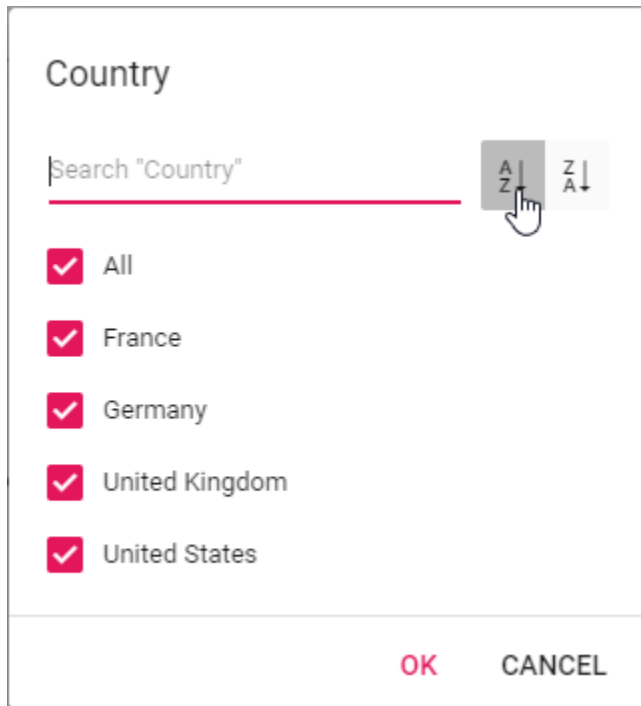
By default, search option is available to quickly navigate to the desired members. It can be done by entering the starting character(s) of the actual members.

A member filter dialog box titled "Year". It features a search bar containing the text "FY 2015" with a clear button (X) to its right. Below the search bar, there is a list of members: "All" and "FY 2015". Each member has a checked checkbox (red square with a white checkmark) to its left. At the bottom of the dialog, there are two buttons: "OK" and "CANCEL". The "OK" button is highlighted in red.



### Option to sort members

User can sort members within the member editor either to ascending (or) descending using the built-in sort icons. When both ascending and descending options are not chosen, then members will be shown in the default order (retrieved as such from data source).



### Performance Tips

In member filter dialog, end user can set the limit to display members while loading large data. Based on this limit, initial loading will get completed quickly without any performance constraint. Also, a message with remaining member count, which are not part of the UI, will be displayed in the member editor.

The data limit can be set using the [MaxNodeLimitInMemberEditor](#) property in [PivotView](#) class. By default, the property holds the numeric value **1000**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .EnableSorting(false).AllowMemberFilter(true)
 }, rows =>
 {
 rows.Name("ProductID").Add();
 }, columns =>
 {
 columns.Name("DeliveryDate").Add();
 }, values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .MaxNodeLimitInMemberEditor(500).ShowGroupingBar(true).Render()
```

### FILTERING.CS

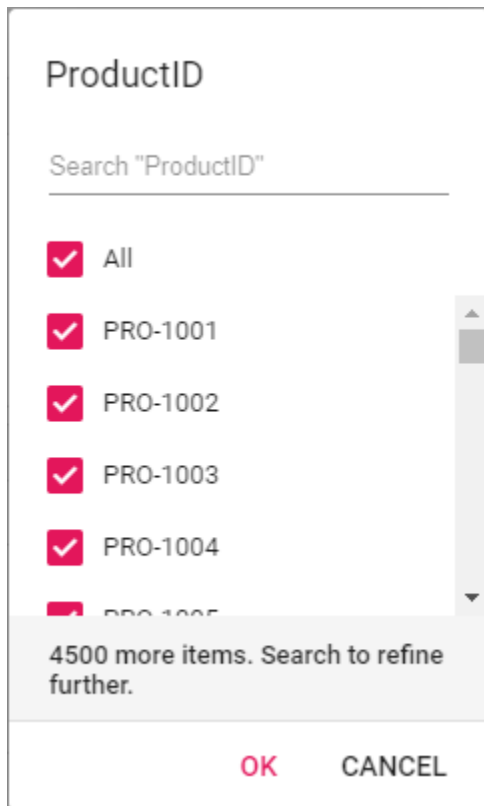
```

public ActionResult Index()
{
 var data = GetProductData();
 ViewBag.DataSource = data;
 ViewBag.filterMembers = new string[] { "United States" };
 return View();
}

public class ProductDetails
{
 public string ProductID { get; set; }
 public int Sold { get; set; }
 public DateTime DeliveryDate { get; set; }
}

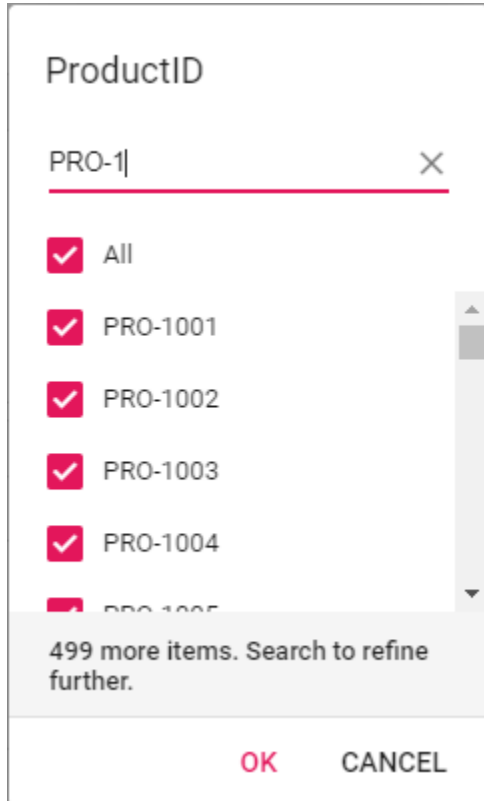
public static List<ProductDetails> GetProductData()
{
 List<ProductDetails> productData = new List<ProductDetails>();
 for (int i = 0; i < 5000; i++)
 {
 int RandomNumber = new Random().Next(1, 10);
 productData.Add(new ProductDetails { Sold = RandomNumber, ProductID
= "PRO-" + (i + 1001), DeliveryDate = new DateTime(2019, 1,
1).AddDays(RandomNumber) });
 }
 return productData;
}

```



Meanwhile, end user can utilize the search option to refine the members from the exceeded limit. For example, consider that there are 5000 members in the name "Node 1", "Node 2", "Node 3", and so on...

and user has set the property [MaxNodeLimitInMemberEditor](#) to **500**. In this case, only the initial 500 members will be displayed by default leaving a message "4500 more items. Search to refine further.". To get the member(s) between 501 to 5000, enter the starting character(s) in search option to bring the desired member(s) from the exceeded limit to the UI. Now, end user can either check or uncheck to continue with the filtering process.



#### [Loading members on-demand](#)

**Note:** This property is applicable only for OLAP data sources.

Allows to load members inside the filter dialog on-demand by setting the [LoadOnDemandInMemberEditor](#) property to **true**. By default, first level is loaded in the member editor from the OLAP cube. So, the member editor will be opened quickly, without any performance constraints. By default, this property is set to **true** and the search will only be applied to the level members that are loaded. In the meantime, the next level members can be added using either of the following methods.

- By clicking on the expander button of the respective member, only its child members will be loaded.
- Select a level from the drop-down list that will load all members up to the chosen level from the cube.

This will help to avoid performance lags when opening a member editor whose hierarchy has a large number of members. Once level members are queried and added one after the other, they will be maintained internally (for all operations like dialog re-opening, drag and drop, etc...) and will not be removed until the web page is refreshed.

**CSHTML**

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").ShowFieldList
(true).ShowGroupingBar(true).LoadOnDemandInMemberEditor(true).AllowCalculate
dField(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); values.Name("Order
on Discount").IsCalculatedField(true).Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
 .CalculatedFieldSettings(calculatedFieldSettings =>
{

calculatedFieldSettings.Name("BikeAndComponents").Formula("([Product].[Produ
ct Categories].[Category].[Bikes] + [Product].[Product
Categories].[Category].[Components])").HierarchyUniqueName("[Product].[Produ
ct Categories]").FormatString("Standard").Add();
 calculatedFieldSettings.Name("Order on
Discount").Formula("[Measures].[Order Quantity] + ([Measures].[Order
Quantity] * 0.10)").FormatString("Currency").Add();
 })).Render()
<style>
 #pivotview {
 display: block;
 }
</style>

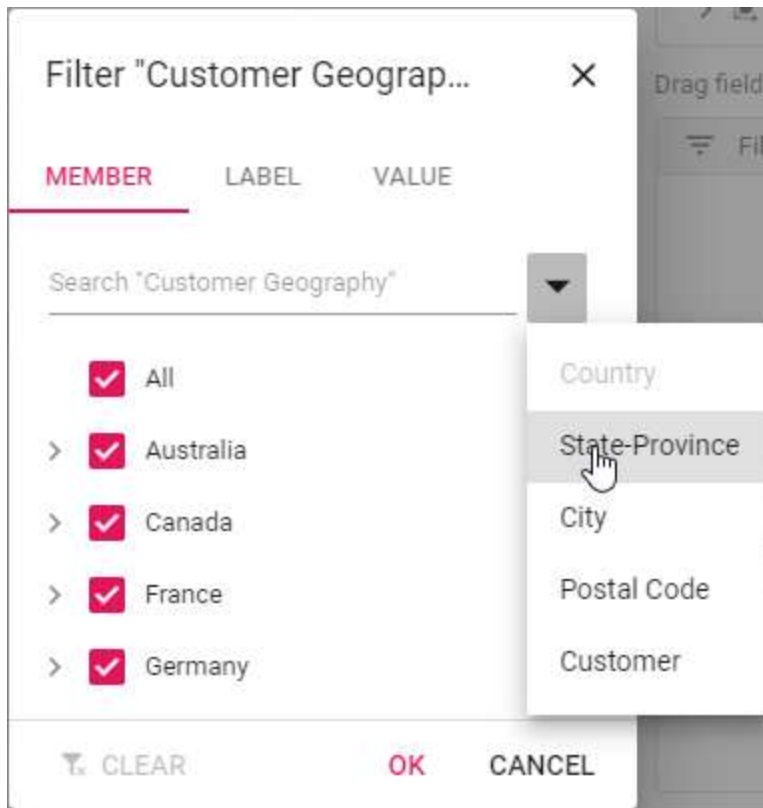
```

**FILTERING.CS**

```

public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal
Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
 return View();
}

```

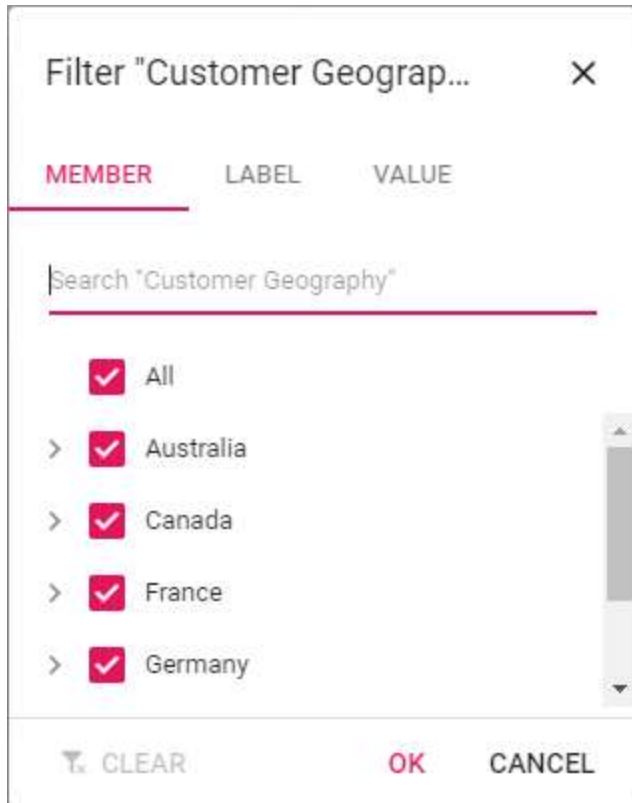


In the example above, "Customer Geography" dimension is loaded with first level (Country) during initial loading. The search will therefore be applied on the members of the "Country" level alone. After that, you can load members to the next level (State-Province) on-demand by expanding the "Australia" node (or) by selecting the "State-Province" level from the drop down list.

- When you expand "Australia", the "State-Province" members will be loaded to "Australia" alone.
- If you load the members by selecting the "State-Province" level from the drop-down list means, the "State-Province" members will be loaded across all countries like Australia, Canada, France, etc.

Once members are loaded, they are maintained internally and will not be removed until the page is refreshed.

If the property is set to **false**, all members of all levels will be queried and added during initial loading itself. Only one query is executed here to retrieve all members from all levels. Since it fetches large number of members, you can feel the performance difference while opening the member editor. But still, expand and search operation is quick here because the members have already been retrieved and populated.



Loading members based on level number

**Note:** This property is applicable only for OLAP data sources.

Allows user to load the members on the basis of the level number set in the [LevelCount](#) property in the [PivotViewFilterSettings](#). By default, this property is set to **1** and the search will only take place within the members of the first level.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("600").ShowFieldList
(true).ShowGroupingBar(true).AllowCalculatedField(true).DataSourceSettings(d
ataSourceSettings => dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
.Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
.Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
.Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); values.Name("Order
on Discount").IsCalculatedField(true).Add(); })
.Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
.FilterSettings(filterSettings =>
{
```

```

 filterSettings.Name("[Customer].[Customer Geography]").Items(ViewBag.filterMembers).LevelCount(2).Add();
 }).CalculatedFieldSettings(calculatedFieldSettings =>
 {

 calculatedFieldSettings.Name("BikeAndComponents").Formula("([Product].[Product Categories].[Category].[Bikes] + [Product].[Product Categories].[Category].[Components])").HierarchyUniqueName("[Product].[Product Categories]").FormatString("Standard").Add();
 calculatedFieldSettings.Name("Order on Discount").Formula("[Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)").FormatString("Currency").Add();
 }).Render()
<style>
 #pivotview {
 display: block;
 }
</style>

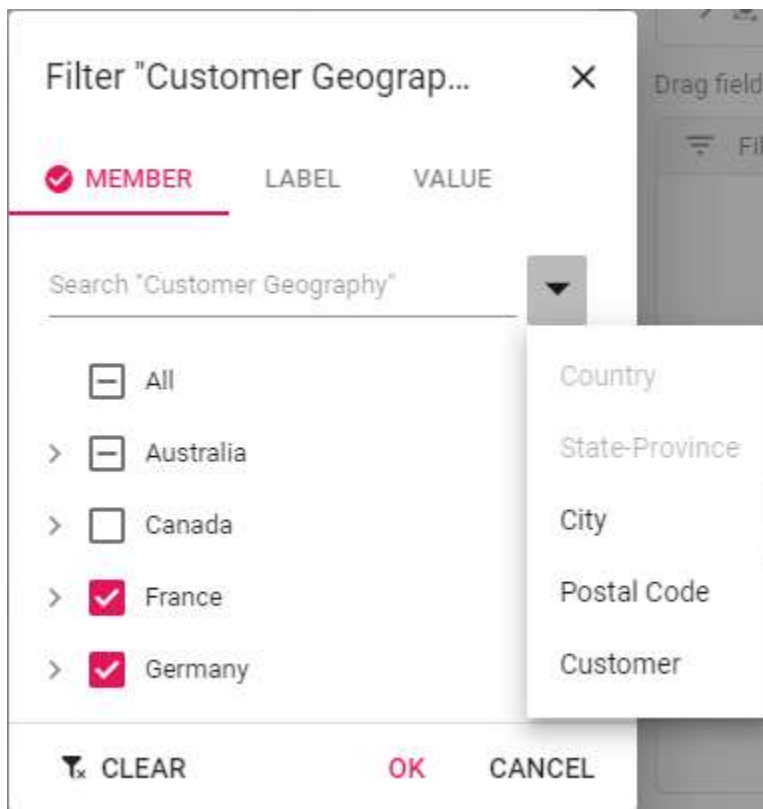
```

**FILTERING.CS**

```

public ActionResult Index()
{
 ViewBag.filterMembers = new string[] { "[Date].[Fiscal].[Fiscal Quarter].&[2002]&[4]", "[Date].[Fiscal].[Fiscal Year].&[2005]" };
 return View();
}

```



In the example above, we set [LevelCount](#) as **2** for the "Customer Geography" dimension in [PivotViewFilterSettings](#). So, the "Customer Geography" dimension is loaded with the "Country" and "State-Province" levels during initial loading itself. The search will therefore be applied only to the members of the "Country" and "State-Province" levels. After that, you can load members to the next level on-demand by expanding the respective "State-Province" node (or) by selecting the "City" level from the drop-down list.

### Label filtering

The label filtering helps to view the pivot table with selective header text in fields across row and column axes based on the applied filter criteria. The following are the three different types of label filtering available:

- Filtering string data type
- Filtering number data type
- Filtering date data type

The label filtering dialog can be enabled by setting the [AllowLabelFilter](#) property in [PivotViewDataSourceSettings](#) class to **true**. After enabling this API, click the filter icon besides any field in row or column axis available in field list or grouping bar UI. Now a filtering dialog will appear and navigate to "Label" tab to perform label filtering operations.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### FILTERING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



Field List

All Fields

Country

Products

Quarter

Sold Amount

Unit Sold

Year

Filters

Drop filter here

Columns

Year

↑

Quarter

↑

Rows

Country

↑

Products

↑

Values

Sum of Unit Sold

▼

Sum of Sold Amount

▼

CLOSE

<br/>

|                        |                          |             |           |
|------------------------|--------------------------|-------------|-----------|
| Sum of Unit Sold ▼ ×   | Drop filter here         |             |           |
| Sum of Sold Amount ▼ × | Year ↑ = × Quarter ↑ = × |             |           |
| Country ↑ = ×          | ► FY 2015                |             | ► FY 2016 |
| Products ↑ = ×         | Unit Sold                | Sold Amount | Unit Sold |
| ► France               | 450                      | \$714,955   | 526       |

<br/>

<br/>

Filter "Country" by ×

MEMBER
LABEL

Show the items for which the label

Does Not Equal ▼

France| ×

✕ CLEAR
OK
CANCEL

&lt;br/&gt;

&lt;br/&gt;

|                        |                          |             |           |             |           |
|------------------------|--------------------------|-------------|-----------|-------------|-----------|
| Sum of Unit Sold ▼ ✕   | Drop filter here         |             |           |             |           |
| Sum of Sold Amount ▼ ✕ | Year ↑ ≡ ✕ Quarter ↑ ≡ ✕ |             |           |             |           |
| Country ↑ ≡ ✕          | FY 2015                  |             | FY 2016   |             |           |
| Products ↑ ≡ ✕         | Unit Sold                | Sold Amount | Unit Sold | Sold Amount | Unit Sold |
| ▶ Germany              | 440                      | \$563,515   | 496       | \$1,772,104 |           |
| ▶ United States        | 546                      | \$754,515   | 636       | \$2,263,104 |           |
| Grand Total            | 986                      | \$1,318,030 | 1132      | \$4,035,208 |           |

**Note:** In label filtering UI, based on the field chosen, it's member data type is automatically recognized and filtering operation will be carried out. Whereas in code behind, user need to define the data type through a property and it has been explained in the immediate section below.

#### Filtering string data type through code

This type of filtering is exclusively applicable for fields with members in string data type. The filtering can be configured using the [PivotViewFilterSettings](#) class through code-behind. The properties required for label filter are:

- [Name](#): Sets the field name.
- [Type](#): Sets the filter type as [FilterType.Label](#) to the field.

- **Condition:** Sets the operator type such as [Operators.Equals](#), [Operators.GreaterThan](#), [Operators.LessThan](#), etc.
- **Value1:** Sets the start value.
- **Value2:** Sets the end value. It is applicable only for the operator such as [Operators.Between](#) and [Operators.NotBetween](#).
- **SelectedField:** Sets level name of a dimension, where the filter settings are to be applied. **This property applicable only for OLAP data source.**

For example, in a "Country" field, to show countries names that contains "United" text alone, set [Value1](#) to "United" and [Condition](#) to [Operators.Contains](#) for desired output in pivot table.

[Operators](#) that can be used in label filtering are:

| Operator             | Description                                                                   |
|----------------------|-------------------------------------------------------------------------------|
| ----- -----          |                                                                               |
| Equals               | Displays the pivot table that matches with the text.                          |
| DoesNotEquals        | Displays the pivot table that does not match with the given text.             |
| BeginWith            | Displays the pivot table that begins with text.                               |
| DoesNotBeginWith     | Displays the pivot table that does not begins with text.                      |
| EndsWith             | Displays the pivot table that ends with text.                                 |
| DoesNotEndsWith      | Displays the pivot table that does not ends with text.                        |
| Contains             | Displays the pivot table that contains text.                                  |
| DoesNotContains      | Displays the pivot table that does not contain text.                          |
| GreaterThan          | Displays the pivot table when the text is greater.                            |
| GreaterThanOrEqualTo | Displays the pivot table when the text is greater than or equal.              |
| LessThan             | Displays the pivot table when the text is lesser.                             |
| LessThanOrEqualTo    | Displays the pivot table when the text is lesser than or equal.               |
| Between              | Displays the pivot table that records between the start and end text.         |
| NotBetween           | Displays the pivot table that does not record between the start and end text. |

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true)
.FilterSettings(filtersettings =>
{
filtersettings.Name("Country").Type(Syncfusion.EJ2.PivotView.FilterType.Label).Condition(Syncfusion.EJ2.PivotView.Operators.Contains)
.Value1("United").Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}
```

```

 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()

```

### **FILTERING.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 546        | \$754,515   | 636        | \$2,263,104 |            |

#### Filtering number data type through code

This type of filtering is exclusively applicable for fields with members in number data type. The filtering can be configured in a similar way explained in the previous section - "Filtering string data type through code", except the [Type](#) property setting. For number data type, set the [Type](#) property to [FilterType.Number](#) enumeration.

For example, in a "Sold" field, to show the values between "90" to "100", set [Value1](#) to "90", [Value2](#) to "100" and [Condition](#) to [Operators.Between](#) for desired output in pivot table.

**Note:** Operators like [Operators.Equals](#), [Operators.DoesNotEquals](#), [Operators.GreaterThan](#), [Operators.GreaterThanOrEqualTo](#), [Operators.LessThan](#), [Operators.LessThanOrEqualTo](#), [Operators.Between](#), and [Operators.NotBetween](#) are alone applicable for number data type.

### **CSHTML**

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true)
 .FilterSettings(filtersettings =>
 {
 filtersettings.Name("Sold").Type(Syncfusion.EJ2.PivotView.FilterType.Number)
 .Condition(Syncfusion.EJ2.PivotView.Operators.Between).Value1("90").Value2(100).Add();
 }).Rows(rows =>
 {
 rows.Name("Sold").Caption("Units Sold").Add();
 }

```

```

 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Amount").Caption("Sold Amount").Add();
 }).Filters(filters =>
 {
 filters.Name("Country").Add(); filters.Name("Products").Add();
 }).Render()

```

### FILTERING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|             | ► FY 2015 | ► FY 20...  | ► FY 20...  | ► FY 20... | Grand To |
|-------------|-----------|-------------|-------------|------------|----------|
| 90          | \$198,808 |             | \$192,220   |            | \$391    |
| 91          | \$67,824  |             |             |            | \$67     |
| 93          | \$139,412 |             |             |            | \$139    |
| 95          |           | \$663,760   | \$837,800   |            | \$1,501  |
| 96          |           |             |             | \$77,264   | \$77     |
| 97          |           | \$1,184,352 |             |            | \$1,184  |
| 99          |           |             | \$2,005,012 |            | \$2,005  |
| Grand Total | \$406,044 | \$1,848,112 | \$3,035,032 | \$77,264   | \$5,366  |

#### [Filtering date data type through code](#)

This type of filtering is exclusively applicable for fields with members in date data type. The filtering can be configured in a similar way explained in the prior section - "Filtering string data type through code", except the [Type](#) property setting. For date data type, set the [Type](#) property to [FilterType.Date](#) enumeration.

For example, in a "Delivery Date" field, to show the records of the the year after 2015, then set [Value1](#) to "2015-01-01" and [Condition](#) to [Operators.After](#) for desired output in pivot table.

**Note:** Operators like [Operators.Equals](#), [Operators.DoesNotEquals](#), [Operators.Before](#), [Operators.BeforeOrEqualTo](#), [Operators.After](#), [Operators.AfterOrEqualTo](#), [Operators.Between](#), and [Operators.NotBetween](#) are alone applicable for date data type.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).Load("onLoad").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true)
 .FilterSettings(filterSettings =>
 {
 filterSettings.Name("Year").Type(Syncfusion.EJ2.PivotView.FilterType.Date).Condition(
 Syncfusion.EJ2.PivotView.Operators.After).Add();
 }).FormatSettings(formatSettings =>
 {
 formatSettings.Name("Year").Format("dd/MM/yyyy-hh:mm").Type("date").Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()
 }
</script>
 function onLoad(args) {
 args.dataSourceSettings.filterSettings[0].value1 = new Date("2015,
1, 1");
 }
</script>
```

### **FILTERING.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                  | Mountain Bikes |             | Road Bikes |             | Grand Total |              |
|------------------|----------------|-------------|------------|-------------|-------------|--------------|
|                  | Units Sold     | Sold Amount | Units Sold | Sold Amount | Units Sold  | Sold Amount  |
| 01/01/2016-12:00 | 794            | \$2,767,656 | 864        | \$2,809,656 | 1658        | \$5,577,312  |
| 01/01/2017-12:00 | 783            | \$2,447,712 | 813        | \$5,131,852 | 1596        | \$7,579,564  |
| 01/01/2018-12:00 | 188            | \$201,792   |            |             | 188         | \$201,792    |
| Grand Total      | 1765           | \$5,417,160 | 1677       | \$7,941,508 | 3442        | \$13,358,668 |

#### Clearing the existing label filter

End user can clear the applied label filter by simply click the "Clear" option at the bottom of the filter dialog under "Label" tab.

### Value filtering

The value filtering helps to perform filter operation based only on value fields and its resultant aggregated values over other fields defined in row and column axes.

The value filtering dialog can be enabled by setting the [AllowValueFilter](#) property in [PivotViewDataSourceSettings](#) class to **true**. After enabling this API, click the filter icon besides any field in row or column axis available in field list or grouping bar UI. Now a filtering dialog will appear and navigate to "Value" tab to perform value filtering operations.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).AllowValueFilter(true)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### FILTERING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

Field List

All Fields

Country

Products

Quarter

Sold Amount

Unit Sold

Year

Filters

Drop filter here

Columns

Year

Quarter

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

CLOSE

<br/>

|                    |                  |             |           |
|--------------------|------------------|-------------|-----------|
| Sum of Unit Sold   | Drop filter here |             |           |
| Sum of Sold Amount | Year    Quarter  |             |           |
| Country            | FY 2015          |             | FY 2016   |
| Products           | Unit Sold        | Sold Amount | Unit Sold |
| France             | 450              | \$714,955   | 526       |

<br/>

<br/>



Filter "Country" by

MEMBER

VALUE

Show the items for which

Unit Sold

Greater Than

1500

CLEAR

OK

CANCEL

&lt;br/&gt;

&lt;br/&gt;

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 996        | \$1,469,470 | 1162       | \$3,805,208 | 1          |

The value filtering can also be configured using the [PivotViewFilterSettings](#) class through code-behind. The properties required for value filter are:

- [Name](#): Sets the normal field name.
- [Type](#): Sets the filter type as [FilterType.Value](#) to the field.
- [Measure](#): Sets the value field name.
- [Condition](#): Sets the operator type such as [Operators.Equals](#), [Operators.GreaterThan](#), [Operators.LessThan](#) etc.
- [Value1](#): Sets the start value.
- [Value2](#): Sets the end value. It is applicable only for the operator such as [Operators.Between](#) and [Operators.NotBetween](#).

For example, to show the data where total sum of units sold for each country exceeding 1500, set [Value1](#) to "1500" and [Condition](#) to [Operators.GreaterThan](#) in the "Country" field.

[Operators](#) that can be used in value filtering are:

| Operator             | Description                                                                 |
|----------------------|-----------------------------------------------------------------------------|
| ----- -----          |                                                                             |
| Equals               | Displays the pivot table that matches with the value.                       |
| DoesNotEquals        | Displays the pivot table that does not match with the given value.          |
| GreaterThan          | Displays the pivot table when the value is greater.                         |
| GreaterThanOrEqualTo | Displays the pivot table when the value is greater than or equal.           |
| LessThan             | Displays the pivot table when the value is lesser.                          |
| LessThanOrEqualTo    | Displays the pivot table when the value is lesser than or equal.            |
| Between              | Displays the pivot table that records between start and end values.         |
| NotBetween           | Displays the pivot table that does not record between start and end values. |

### C#HTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).AllowValueFilter(true)
.FilterSettings(filtersettings =>
{
filtersettings.Name("Country").Measure("Sold").Type(Syncfusion.EJ2.PivotView
.FilterType.Value).Condition(Syncfusion.EJ2.PivotView.Operators.GreaterThan)
.Value1("1500").Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### FILTERING.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```



```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).MemberFiltering("memberFiltering").Render()
<script>
 function memberFiltering(args) {
 args.cancel = true;
 }
</script>
```

### FILTERING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.filterMembers = new string[] { "FY 2017" };
 return View();
}
```

### MemberEditorOpen

The event [MemberEditorOpen](#) fires while opening member editor dialog. It allows to customize the field members to be displayed in the dialog. It has the following parameters

- **fieldName**: It holds the name of the appropriate field.
- **fieldMembers**: It holds the members of a field.
- **cancel**: It is a boolean property and by setting this to true, dialog won't be created.

In the below sample, the member editor of field "Country" shows only the selected Item.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
```

```

 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).MemberEditorOpen("memberEditorOpen").Render()
<script>
 function memberEditorOpen(args) {
 if (args.fieldName == 'Country') {
 args.fieldMembers = args.fieldMembers.filter((key) => {
 return (key.actualText == 'France' || key.actualText ==
'Germany')
 });
 }
 }
</script>

```

### **FILTERING.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.filterMembers = new string[] { "FY 2017" };
 return View();
}

```

### ActionBegin

The event [actionBegin](#) triggers when clicking the filter icon in the field button, which is present in both grouping bar and field list UI. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. For example, while filtering, the action name will be shown as **Filter field**.
- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel**: It allows user to restrict the current action.

In the below sample, filter action can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

### **CSHTML**

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {

```

```

AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Filter field') {
 args.cancel = true;
 }
 }
</script>

```

### ACTIONBEGIN-AGGREGATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### ActionComplete

The event [actionComplete](#) triggers when the filtering action via the field button, which is present in both grouping bar and field list UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. For example, after filtering, the action name will be shown as **Field filtered**.
- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **actionInfo**: It holds the unique information about the current UI action. For example, if the filter action is completed, the event argument contains information such as filter members, field name, and so on.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
```

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource) .ExpandAll (fal
se) .EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'Field filtered') {
 // Triggers when the filter action is completed.
 }
 }
</script>
```

### ACTIONCOMPLETE-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. For example, if the action fails while filtering, the action name will be shown as **Filter field**.
- **errorInfo**: It holds the error information of the current UI action.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource) .ExpandAll (fal
se) .EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
```

```

columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowGroupingBar(true).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Filter field') {
 // Triggers when the current UI action fails to achieve the
 desired result.
 }
 }
</script>

```

### **ACTIONFAILURE-AGGREGATION.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

## Sorting in ASP.NET MVC Pivot Table Component

### *Member Sorting*

Allows to order field members in rows and columns either in ascending or descending order. By default, field members in rows and columns are in ascending order.

Member sorting can be enabled by setting the [EnableSorting](#) property in [PivotViewDataSourceSettings](#) class to **true**. After enabling this API, click the sort icon besides each field in row or column axis, available in field list or grouping bar UI for re-arranging members either in ascending or descending order.

**Note:** By default the [EnableSorting](#) property in [PivotViewDataSourceSettings](#) class set as **true**. If we set it as **false**, then the field members arrange in pivot table as its data source order. And, the sort icons in grouping bar and field list buttons will be removed.



Field List

All Fields

Country

Products

Quarter

Sold Amount

Unit Sold

Year

Filters

Drop filter here

Columns

Year

Quarter

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

CLOSE

<br/>

|                    |                  |             |           |             |           |
|--------------------|------------------|-------------|-----------|-------------|-----------|
| Sum of Unit Sold   | Drop filter here |             |           |             |           |
| Sum of Sold Amount | YearQuarter      |             |           |             |           |
| Country            | FY 2015          |             | FY 2016   |             |           |
| Products           | Unit Sold        | Sold Amount | Unit Sold | Sold Amount | Unit Sold |
| United States      | 546              | \$754,515   | 636       | \$2,263,104 |           |
| Germany            | 440              | \$563,515   | 496       | \$1,772,104 |           |
| France             | 450              | \$714,955   | 526       | \$1,542,104 |           |
| Grand Total        | 1436             | \$2,032,985 | 1658      | \$5,577,312 |           |

<br/>

|               | FY 2015   |             | FY 2016   |             | FY 2017   |
|---------------|-----------|-------------|-----------|-------------|-----------|
|               | Unit Sold | Sold Amount | Unit Sold | Sold Amount | Unit Sold |
| United States | 546       | \$754,515   | 636       | \$2,263,104 |           |
| Germany       | 440       | \$563,515   | 496       | \$1,772,104 |           |
| France        | 450       | \$714,955   | 526       | \$1,542,104 |           |
| Grand Total   | 1436      | \$2,032,985 | 1658      | \$5,577,312 | 1         |

Member sorting can also be configured using the [PivotViewSortSettings](#) class through code behind, during initial rendering. The settings required to sort are:

- [Name](#): It allows to set the field name.
- [Order](#): It allows to set the sort direction either to ascending or descending of the respective field.

**Note:** By default the [Order](#) property in the [PivotViewSortSettings](#) class set as [Sorting.Ascending](#). Meanwhile, we can arrange the field members as its order in data source by setting it as [Sorting.None](#) where the sort icons in grouping bar and field list buttons for the corresponding field will be removed.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.SortSettings(sortsettings =>
{
 sortsettings.Name("Country").Order(Sorting.Descending).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### SORTING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Alphanumeric Sorting

Usually string sorting is applied to field members even if it starts with numbers. But this kind of field members can also be sorted on the basis of numbers that are placed at the beginning of the member name. This can be achieved by setting the [dataType](#) property as **number** to the desired field.

### CSHTML

```
@Html.EJS().PivotView("PivotGrid").Width("100%").Height("300").DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)
 ViewBag.Data)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Country").Add();
 })
 .Rows(rows =>
 {
 rows.Name("ProductID").DataType("Number").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 .ShowGroupingBar(true).Render()
 })
```

### ALPHAHEADER.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { ProductID = "618-XW", Country = "Canada",
 Sold = 90, Amount = 9219069 });
 pivotData.Add(new PivotData { ProductID = "1111-GQ", Sold = 37, Amount =
 1571126, Country = "Australia" });
 pivotData.Add(new PivotData { ProductID = "330-BR", Sold = 31, Amount =
 9523258, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "1035-VC", Sold = 86, Amount =
 1004572, Country = "United States" });
 pivotData.Add(new PivotData { ProductID = "36-SW", Sold = 73, Amount =
 4532163, Country = "United Kingdom" });
 pivotData.Add(new PivotData { ProductID = "71-AJ", Sold = 45, Amount =
 1916052, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "980-PP", Sold = 85, Amount =
 6586156, Country = "Canada" });
}
```

```

 pivotData.Add(new PivotData { ProductID = "209-FB", Sold = 51, Amount =
6348087, Country = "Australia" });
 pivotData.Add(new PivotData { ProductID = "428-PL", Sold = 65, Amount =
1365854, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "618-XW", Sold = 81, Amount =
6461768, Country = "United States" });
 pivotData.Add(new PivotData { ProductID = "1111-GQ", Sold = 33, Amount =
6181560, Country = "United Kingdom" });
 pivotData.Add(new PivotData { ProductID = "330-BR", Sold = 17, Amount =
611364, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "1035-VC", Sold = 41, Amount =
3688930, Country = "Canada" });
 pivotData.Add(new PivotData { ProductID = "36-SW", Sold = 51, Amount =
4648920, Country = "Australia" });
 pivotData.Add(new PivotData { ProductID = "71-AJ", Sold = 56, Amount =
4579862, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "980-PP", Sold = 25, Amount =
1249117, Country = "United States" });
 pivotData.Add(new PivotData { ProductID = "209-FB", Sold = 60, Amount =
9603891, Country = "United Kingdom" });
 pivotData.Add(new PivotData { ProductID = "428-PL", Sold = 31, Amount =
9548655, Country = "Canada" });
 pivotData.Add(new PivotData { ProductID = "618-XW", Sold = 93, Amount =
7496742, Country = "Australia" });
 pivotData.Add(new PivotData { ProductID = "1111-GQ", Sold = 62, Amount =
8692814, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "330-BR", Sold = 22, Amount =
4789234, Country = "United States" });
 pivotData.Add(new PivotData { ProductID = "1035-VC", Sold = 61, Amount =
7927531, Country = "United Kingdom" });
 pivotData.Add(new PivotData { ProductID = "36-SW", Sold = 68, Amount =
5440025, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "71-AJ", Sold = 87, Amount =
8097913, Country = "Canada" });
 pivotData.Add(new PivotData { ProductID = "980-PP", Sold = 87, Amount =
1809071, Country = "Australia" });
 pivotData.Add(new PivotData { ProductID = "209-FB", Sold = 96, Amount =
9893092, Country = "Germany" });
 pivotData.Add(new PivotData { ProductID = "428-PL", Sold = 22, Amount =
8136252, Country = "United States" });
 pivotData.Add(new PivotData { ProductID = "618-XW", Sold = 29, Amount =
9190577, Country = "United Kingdom" });
 pivotData.Add(new PivotData { ProductID = "1111-GQ", Sold = 85, Amount =
5410172, Country = "Germany" });
 return pivotData;
 }
}

public class PivotData
{
 public int Sold { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string ProductID { get; set; }
}

```

| ProductID | Australia  |             | Canada     |             | Germany    |             | United Kingdom |             | United States |
|-----------|------------|-------------|------------|-------------|------------|-------------|----------------|-------------|---------------|
|           | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold     | Sold Amount | Units Sold    |
| 35-BW     | 51         | 4648920     |            |             | 66         | 5440029     | 73             | 4532163     |               |
| 71-AJ     |            |             | 87         | 6097913     | 101        | 6495914     |                |             |               |
| 209-FB    | 51         | 6348067     |            |             | 96         | 9593092     | 60             | 9603891     |               |
| 330-BR    |            |             |            |             | 48         | 10134622    |                |             |               |
| 428-PL    |            |             | 31         | 9548655     | 65         | 1365854     |                |             |               |
| 618-XW    | 93         | 7496742     | 90         | 9219069     |            |             | 29             | 9190577     |               |
| 980-PP    | 87         | 1809071     | 85         | 6586106     |            |             |                |             |               |

### Custom Sorting

Allows to sort field headers (aka, members) in rows and columns based on user-defined order. This can be configured mainly using the [MembersOrder](#) in the [PivotViewSortSettings](#) through code behind, during initial rendering. The other settings required to sort are:

- [Name](#) : It allows to set the field name.
- [MembersOrder](#) : It holds an array of headers in the order specified by the user.
- [Order](#) : It allows to specify whether the array of headers should be sorted ascending or descending.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.SortSettings(sortsettings =>
{
sortsettings.Name("Country").Order(Sorting.Ascending).MembersOrder(ViewBag.membersOrder).Add();
sortsettings.Name("Year").Order(Sorting.Descending).MembersOrder(ViewBag.membersOrder_1).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Production Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).Render()
```

### CUSTOM-SORTING.CS

```
public ActionResult Index()
{
var data = GetPivotData();
```

```

ViewBag.DataSource = data;
ViewBag.membersOrder = new string[] { "United Kingdom", "France" };
ViewBag.membersOrder_1 = new string[] { "FY 2015", "FY 2017" };
return View();
}

```

![output](images/Custom sorting.png "Custom Sorting")

### Value Sorting

**Note:** This property is applicable only for relational data source.

Allows to sort individual value field and its aggregated values either in row or column axis in both ascending and descending order. It can be enabled by setting the [EnableValueSorting](#) property in [PivotView](#) class to **true**. On enabling, end user can sort the values by directly clicking the value field header positioned either in row or column axis of the pivot table component.

The value sorting can also be configured using the [PivotViewValueSortSettings](#) option through code behind. The settings required to sort value fields are:

- [HeaderText](#): It allows to set the header names with delimiters, that is used for value sorting. The header names are arranged from Level 1 to Level N, down the hierarchy with a delimiter for better specification.
- [HeaderDelimiter](#): It allows to set the delimiters string to separate the header text between levels.
- [SortOrder](#): It allows to set the sort direction of the value field.

**Note:** Value fields are set to the column axis by default. In such cases, the value sorting applied will have an effect on the column alone. You need to place the value fields in the row axis to do so in row wise. For more information, [refer here](#).

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.ValueSortSettings(new PivotViewValueSortSettings {
 HeaderText = "FY 2015##Sold Amount",
 HeaderDelimiter = "##",
 SortOrder = Sorting.Descending })
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.EnableValueSorting(true).Render()

```

**VALUESORTING.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |               | FY 2016    |             | FY 2017    |
|---------------|------------|---------------|------------|-------------|------------|
|               | Units Sold | Sold Amount ↓ | Units Sold | Sold Amount | Units Sold |
| United States | 546        | \$754,515     | 636        | \$2,263,104 |            |
| France        | 450        | \$714,955     | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515     | 496        | \$1,772,104 |            |
| Grand Total   | 1436       | \$2,032,985   | 1658       | \$5,577,312 | 1          |

*Event**OnHeadersSort*

The event [OnHeadersSort](#) triggers when clicking the value sort icon or the sort icon in the field button, which is present in both grouping bar and field list UI. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **fieldName**: It holds the field name where the sort settings applied.
- **sortOrder**: It holds the current sort order of the field.
- **members**: It holds the sorted headers according to the specified sort order.
- **levelName**: It holds the specific field's unique level name. **Note**: This option is applicable only for OLAP data.
- **isOrderChanged**: By setting this boolean property to true, it allows to display the modified members order.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Production Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
)
```

```

 }).ShowGroupingBar(true).OnHeadersSort("onHeadersSort").Render()
<script>
 function onHeadersSort(args) {
 if (args.fieldName == 'Country') {
 args.members = ['United Kingdom', 'Germany'];
 args.IsOrderChanged = true;
 }
 if (args.fieldName == 'Year') {
 args.members = ['FY 2017', 'FY 2015'];
 args.IsOrderChanged = true;
 }
 }
</script>

```

### ONHEADERSORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

![output](images/Custom sorting\_event.png "Custom Sorting Event")

#### ActionBegin

The event [actionBegin](#) triggers when clicking the value sort icon or the sort icon in the field button, which is present in both grouping bar and field list UI. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

| Action                          | Action Name |
|---------------------------------|-------------|
| -----                           | -----       |
| <a href="#">Sort field</a>      | Sort field  |
| <a href="#">Value sort icon</a> | Sort value  |

- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel**: It allows user to restrict the current action.

In the below sample, sort action can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.



**CSHTML**

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = true
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Sort field' || args.actionName == 'Sort
value') {
 args.cancel = true;
 }
 }
</script>

```

**ACTIONBEGIN-AGGREGATION.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

**ActionComplete**

The event [actionComplete](#) triggers when the UI actions such as value sorting or sorting via the field button, which is present in both grouping bar and field list UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action                          | Action Name  |
|---------------------------------|--------------|
|                                 |              |
| <a href="#">Sort field</a>      | Field sorted |
| <a href="#">Value sort icon</a> | Value sorted |

- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **actionInfo**: It holds the unique information about the current UI action. For example, if sorting is completed, the event argument contains information such as sort order and the field name.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = true
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'Field sorted' || args.actionName == 'Value
sorted') {
 // Triggers when the sort action is completed.
 }
 }
</script>
```

### ACTIONCOMPLETE-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

| Action | Action Name |

|-----|-----|

| [Sort field](#) | Sort field |

| [Value sort icon](#) | Sort value |

- **errorInfo**: It holds the error information of the current UI action.

## CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Sort field' || args.actionName == 'Sort
value') {
 // Triggers when the current UI action fails to achieve the
desired result.
 }
 }
}</script>
```

## ACTIONFAILURE-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

<!-- markdownlint-disable MD012 -->

### Drill Through

Allows to view the underlying raw data of a summarized cell in the pivot table. It can be enabled by setting the [AllowDrillThrough](#) property to **true**. By double-clicking on any value cell, user can view the detailed raw data in a data grid inside a new window. In the new window, row header, column header

and measure name of the clicked cell will be shown at the top. Also, user can include or exclude fields available in the data grid using column chooser option.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").ShowTooltip(false).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowDrillThrough(true).Render()
```

### DRILLTHROUGH.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

|                   | FY 2015    |             | FY 2016    |                 | FY 2017    |                 | FY 2018    |             |
|-------------------|------------|-------------|------------|-----------------|------------|-----------------|------------|-------------|
|                   | Units Sold | Sold Amount | Units Sold | Sold Amount     | Units Sold | Sold Amount     | Units Sold | Sold Amount |
| France            | 46939      | 19097073.84 | 44513      | 19575092.52     | 48207      | 18251898.21     | 11516      |             |
| Bottles and Cages | 3520       | 28019.75    | 3770       | 29833.5         | 3564       | 28398.5         | 529        |             |
| Cleaners          |            |             | 207        | 14250           | 4072       | 17738           | 734        |             |
| Fenders           |            |             | 607        | 41553           | 3589       | 40639.5         | 1035       |             |
| Gloves            |            |             | 2997       | 19151           | 3716       | 24531.5         | 776        |             |
| Helmets           | 3453       | 54174.09    | 3259       | 50947.269999... | 3432       | 53738.709999... | 594        |             |
| Hydration Packs   | 3693       | 113150.75   | 3512       | 107698          | 3412       | 104335          | 897        |             |
| Jerseys           | 3183       | 90561       | 3463       | 98283           | 4187       | 118608.25       | 1190       |             |
| Mountain Bikes    | 4036       | 6880266     | 3972       | 6769705         | 3791       | 6459305         | 1126       |             |
| Road Bikes        | 3729       | 6258897     | 3865       | 5922311         | 4263       | 5605662         | 1010       |             |
| Shorts            | 4740       | 78679.25    | 2952       | 49805           | 3477       | 57933           | 790        |             |
| Tires and Tubes   | 3433       | 66513.25    | 3328       | 64507.75        | 3979       | 77030           | 558        |             |
| Touring Bikes     | 3204       | 5111308.25  | 3891       | 6207461.75      | 3368       | 5373243.75      | 1060       |             |
| Vests             | 3360       | 332910.5    | 2690       | 199586.25       | 3357       | 290735          | 1217       |             |

&lt;br/&gt;

&lt;br/&gt;

|                   | FY 2015    |             | FY 2016    |                 | FY 2017    |                 | FY 2018    |             |
|-------------------|------------|-------------|------------|-----------------|------------|-----------------|------------|-------------|
|                   | Units Sold | Sold Amount | Units Sold | Sold Amount     | Units Sold | Sold Amount     | Units Sold | Sold Amount |
| France            | 46939      | 19097073.84 | 44513      | 19575092.52     | 48207      | 18251898.21     | 11516      |             |
| Bottles and Cages | 3520       | 28019.75    | 3770       | 29833.5         | 3564       | 28398.5         | 529        |             |
| Cleaners          |            |             | 207        | 14250           | 4072       | 17738           | 734        |             |
| Fenders           |            |             | 607        | 41553           | 3589       | 40639.5         | 1035       |             |
| Gloves            |            |             | 2997       | 19151           | 3716       | 24531.5         | 776        |             |
| Helmets           | 3453       | 54174.09    | 3259       | 50947.269999... | 3432       | 53738.709999... | 594        |             |
| Hydration Packs   | 3693       | 113150.75   | 3512       | 107698          | 3412       | 104335          | 897        |             |
| Jerseys           | 3183       | 90561       | 3463       | 98283           | 4187       | 118608.25       | 1190       |             |
| Mountain Bikes    | 4036       | 6880266     | 3972       | 6769705         | 3791       | 6459305         | 1126       |             |
| Road Bikes        | 3729       | 6258897     | 3865       | 5922311         | 4263       | 5605662         | 1010       |             |
| Shorts            | 4740       | 78679.25    | 2952       | 49805           | 3477       | 57933           | 790        |             |
| Tires and Tubes   | 3433       | 66513.25    | 3328       | 64507.75        | 3979       | 77030           | 558        |             |
| Touring Bikes     | 3204       | 5111308.25  | 3891       | 6207461.75      | 3368       | 5373243.75      | 1060       |             |
| Vests             | 3360       | 332910.5    | 2690       | 199586.25       | 3357       | 290735          | 1217       |             |

| Details                                                                       |         |                 |         |             |            |
|-------------------------------------------------------------------------------|---------|-----------------|---------|-------------|------------|
| Row: France - Bottles and Cages    Column: FY 2015    Sum of Units Sold: 3520 |         |                 |         |             |            |
| Quarter                                                                       | Year    | Products        | Country | Sold Amount | Units Sold |
| Q3                                                                            | FY 2015 | Bottles and ... | France  | 3136.5      | 369        |
| Q3                                                                            | FY 2015 | Bottles and ... | France  | 3177.5      | 410        |
| Q3                                                                            | FY 2015 | Bottles and ... | France  | 742.5       | 99         |
| Q4                                                                            | FY 2015 | Bottles and ... | France  | 375.5       | 50         |
| Q4                                                                            | FY 2015 | Bottles and ... | France  | 1096.5      | 129        |
| Q4                                                                            | FY 2015 | Bottles and ... | France  | 3131        | 404        |
| Q4                                                                            | FY 2015 | Bottles and ... | France  | 990         | 132        |

Users can also view the underlying raw data though the pivot chart. By clicking on any data point, user can view the detailed raw data in a data grid inside a new window.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").ShowTooltip(false).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true))
```

```

.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowDrillThrough(true).DisplayOption(new PivotViewDisplayOption { View = View.Chart }).ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()

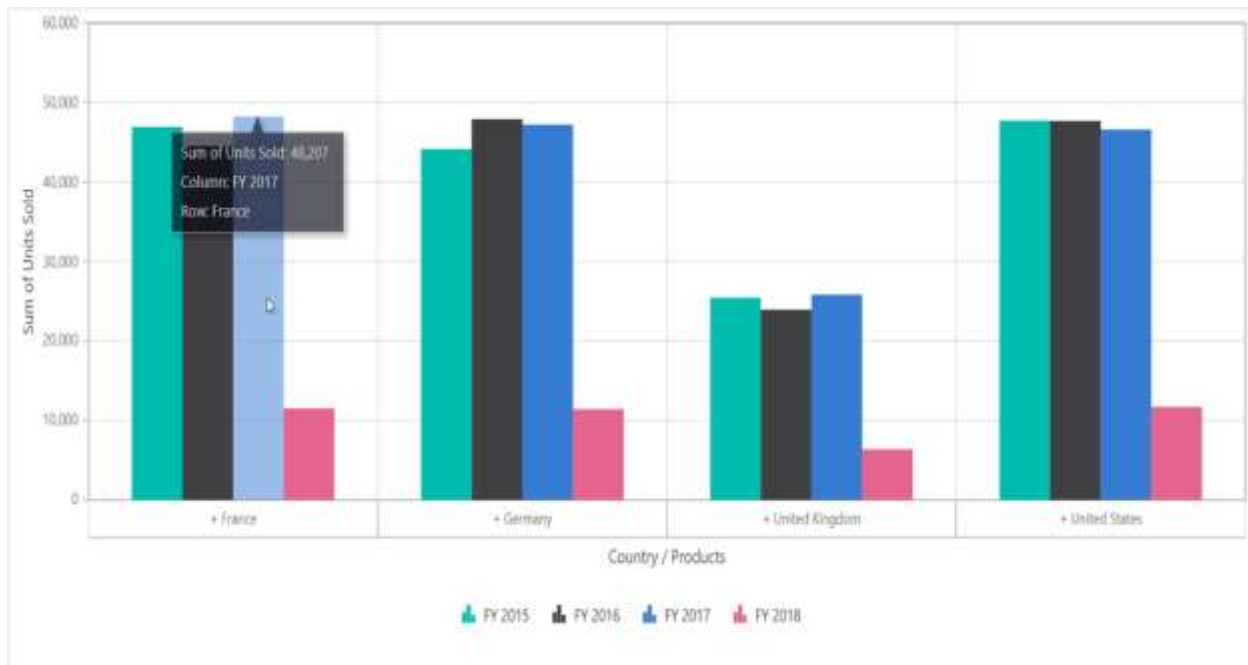
```

### DRILLTHROUGHCHART.CS

```

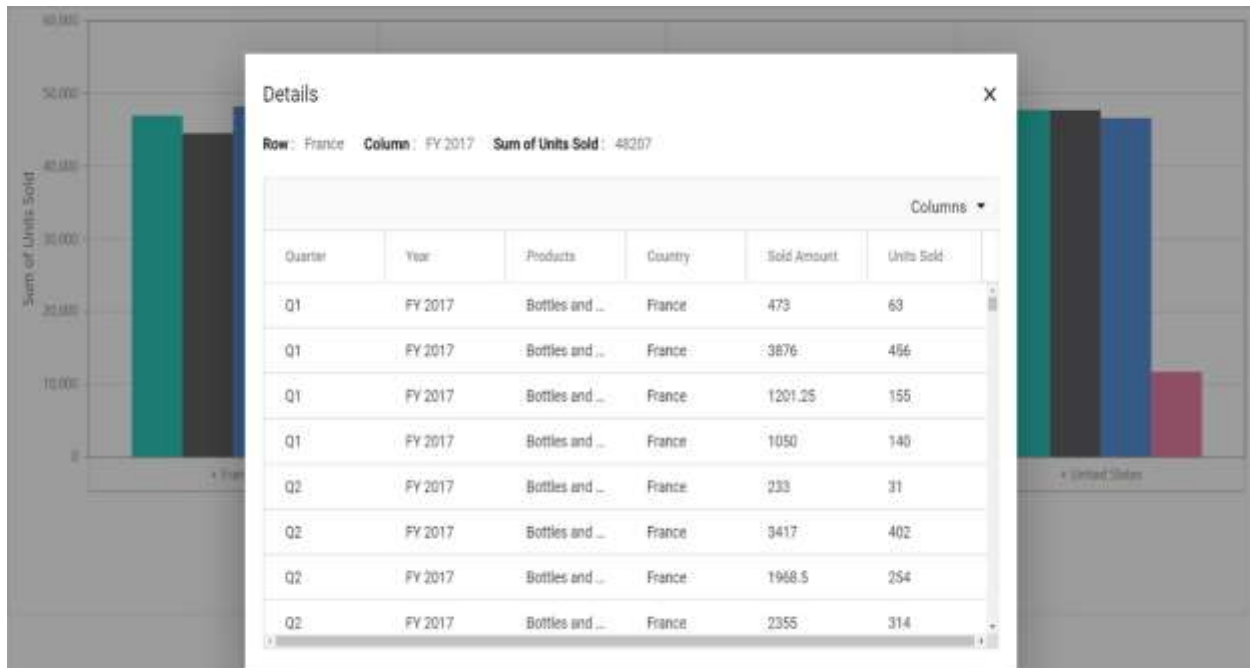
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



&lt;br/&gt;

&lt;br/&gt;



*Maximum rows to retrieve*

**Note:** This property is applicable only for OLAP data sources.

The [MaxRowsInDrillThrough](#) property allows to specify the maximum number of raw data to be returned during the drill through process. By default, this property is set to **"10000"** meaning that if you do not specify this property, you will get 10,000 or less raw data.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("600").ShowFieldList
(true).ShowGroupingBar(true).AllowDrillThrough(true).MaxRowsInDrillThrough(1
0).AllowCalculatedField(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.EnableSorting(true)

.Url("https://bi.syncfusion.com/olap/msmdpump.dll").Catalog("Adventure Works
DW 2008 SE").Cube("Adventure Works").ProviderType(ProviderType.SSAS)
 .Rows(rows => { rows.Name("[Customer].[Customer
Geography]").Caption("Customer Geography").Add(); })
 .Columns(columns => { columns.Name("[Product].[Product
Categories]").Caption("Product Categories").Add();
columns.Name("[Measures]").Caption("Measures").Add(); })
 .Values(values => { values.Name("[Measures].[Customer
Count]").Caption("Customer Count").Add(); values.Name("[Measures].[Internet
Sales Amount]").Caption("Internet Sales Amount").Add(); values.Name("Order
on Discount").IsCalculatedField(true).Add(); })
 .Filters(filters => { filters.Name("[Date].[Fiscal]").Caption("Date
Fiscal").Add(); })
 .FilterSettings(filterSettings =>
 {
filterSettings.Name("[Date].[Fiscal]").Items(ViewBag.filterMembers).LevelCou
nt(3).Add();
 }).CalculatedFieldSettings(calculatedFieldSettings =>
```

```

{
 calculatedFieldSettings.Name("BikeAndComponents").Formula("([Product].[Product Categories].[Category].[Bikes] + [Product].[Product Categories].[Category].[Components])").HierarchyUniqueName("[Product].[Product Categories]").FormatString("Standard").Add();
 calculatedFieldSettings.Name("Order on Discount").Formula("[Measures].[Order Quantity] + ([Measures].[Order Quantity] * 0.10)").FormatString("Currency").Add();
 })).Render()
<style>
 #pivotview {
 display: block;
 }
</style>

```

### DRILL-THROUGH.CS

```

public ActionResult Index()
{
 return View();
}

```

Details

Row: FY 2016 Column: Australia--New South Wales--Coffs Harbour Sum of Internet Sales Amount: \$39,856.79

Drag a column header here to group its column:

| Item... | Item... | Item...  | Item... | Item... | Item...   | Item...   | Costs...       | DateO...       |
|---------|---------|----------|---------|---------|-----------|-----------|----------------|----------------|
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Ariana Stew... | January 29...  |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Ariana Stew... | January 29...  |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Suzanne K. ... | February 3...  |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Suzanne K. ... | February 3...  |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Micah Wu       | March 8, 20... |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Micah Wu       | March 8, 20... |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Jasmine Wl...  | March 23, 2... |
| 3578.27 | 3578.27 | 286.2616 | 89.4568 | 3578.27 | 2171.2942 | 2171.2942 | Jasmine Wl...  | March 23, 2... |

### Events

#### DrillThrough

The event `DrillThrough` triggers every time before a value cell is double clicked. This event allows user to customize the data grid columns in drill through popup. Exclusively the event helps to view and process the raw data information behind a aggregated value inside value cell. It has the following parameters:

- `columnHeaders` - It holds column header of the current cell.
- `currentCell` - It holds the current cell's information.



- `currentTarget` - It holds current cell's html element.
- `gridColumns` - It holds data grid columns to be rendered in drill through popup.
- `rowData` - It holds current cell's raw data.
- `rowHeaders` - It holds row header of current cell.
- `value` - It holds value of current cell.
- `cancel` - It is a boolean property and by setting this to true, dialog won't be created.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").ShowTooltip(false).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).AllowDrillThrough(true).DrillThrough("drillThrough").Render()
<script>
function drillThrough(args) {
//triggers on value cell double click
}
</script>
```

### DRILLTHROUGHEVENT.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

### BeginDrillThrough

The event `BeginDrillThrough` occurs for each and every value cell with a double click, and the event argument provides the data grid information before the drill-through popup is shown. User can access the data grid (which holds the raw data underneath the aggregated value cell) options such as sort, group, filter and customize those in the data grid. It has the following parameters:

- `gridObj` - It holds the data grid instance to be rendered inside the drill-through popup.

- **cellInfo** - It holds current cell information like raw data, row header, column header and value.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").ShowTooltip(false).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.PivotView.EditMode.Normal)).BeginDrillThrough("beginDrillThrough").Render()
<script>
function beginDrillThrough(args) {
 if (args.gridObj) {
 var gridObj = args.gridObj;
 gridObj.allowGrouping = true;
 gridObj.allowSorting = true;
 gridObj.allowFiltering = true;
 gridObj.filterSettings = { type: 'CheckBox' };
 }
}
</script>
```

### CUSTOMEDITING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Editing

**Note:** This feature is applicable only for relational data source.

Cell edit allows to add, delete, or update the raw items of any value cell from the pivot table. The raw items can be viewed in a data grid inside a new window on double-clicking the appropriate value cell. In

the data grid, CRUD operations can be performed by double-clicking the cells or using toolbar options. Once user finishes editing raw items, aggregation will be performed for the updated values in pivot table component immediately. This support can be enabled by setting the [AllowEditing](#) property in [PivotViewCellEditSettings](#) class to **true**.

The CRUD operations available in the data grid toolbar and command column are:

```
| Toolbar Button | Actions |
|-----|-----|
| Add | Add a new row. |
| Edit | Edit the current row or cell. |
| Delete | Delete the current row. |
| Update | Update the edited row or cell. |
| Cancel | Cancel the edited state. |
```

The following are the supported edit types in the data grid:

- Normal
- Dialog
- Batch
- Command Columns

#### *Normal*

In normal edit mode, when user starts editing, the state of the currently selected row alone will be completely changed to edit state. User can change the cell values and save it to the data source by clicking "Update" toolbar button. To enable the normal edit, set the [Mode](#) property in [PivotViewCellEditSettings](#) class to [EditMode.Normal](#).

**Note:** The normal edit mode is the default mode of editing..

#### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
 }).Values(values =>
 {
```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();

 })).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.PivotView.EditMode.Normal)).Render()

```

### NORMAL.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015   | FY 2016     | FY 2017     | FY 2018  | Grand Total |
|---------------|-----------|-------------|-------------|----------|-------------|
| France        | \$714,955 | \$1,542,104 | \$2,903,308 | \$27,264 | \$5,187,631 |
| Germany       |           |             |             |          |             |
| United States |           |             |             |          |             |
| Grand Total   |           |             |             |          |             |

Details

Row: France Column: FY 2015 Sum of Sold Amount: \$714,955

+

 Add
 

✎

 Edit
 

✖

 Delete
 

🔄

 Update
 

✕

 Cancel

Columns ▾

| Quarter | Year    | Products       | Country | Sold Amount |
|---------|---------|----------------|---------|-------------|
| Q1      | FY 2015 | Mountain Bikes | France  | 52,824.00   |
| Q2      | FY 2015 | Mountain Bikes | France  | 86904       |
| Q3      | FY 2015 | Mountain Bikes | France  | 153360      |
| Q4      | FY 2015 | Mountain Bikes | France  | 42600       |
| Q1      | FY 2015 | Road Bikes     | France  | 124422      |
| Q2      | FY 2015 | Road Bikes     | France  | 85448       |

### Dialog

In dialog edit mode, when you start editing, the currently selected row data will be shown in a dialog.

You can change the cell values and save it to the data source by clicking **Save**.

To enable the dialog edit, set the [Mode](#) property in [PivotViewCellEditSettings](#) class to [EditMode.Dialog](#).

### CSHTML

```

@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }
)
)

```

```

 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.PivotView.EditMode.Dialog)).Render()

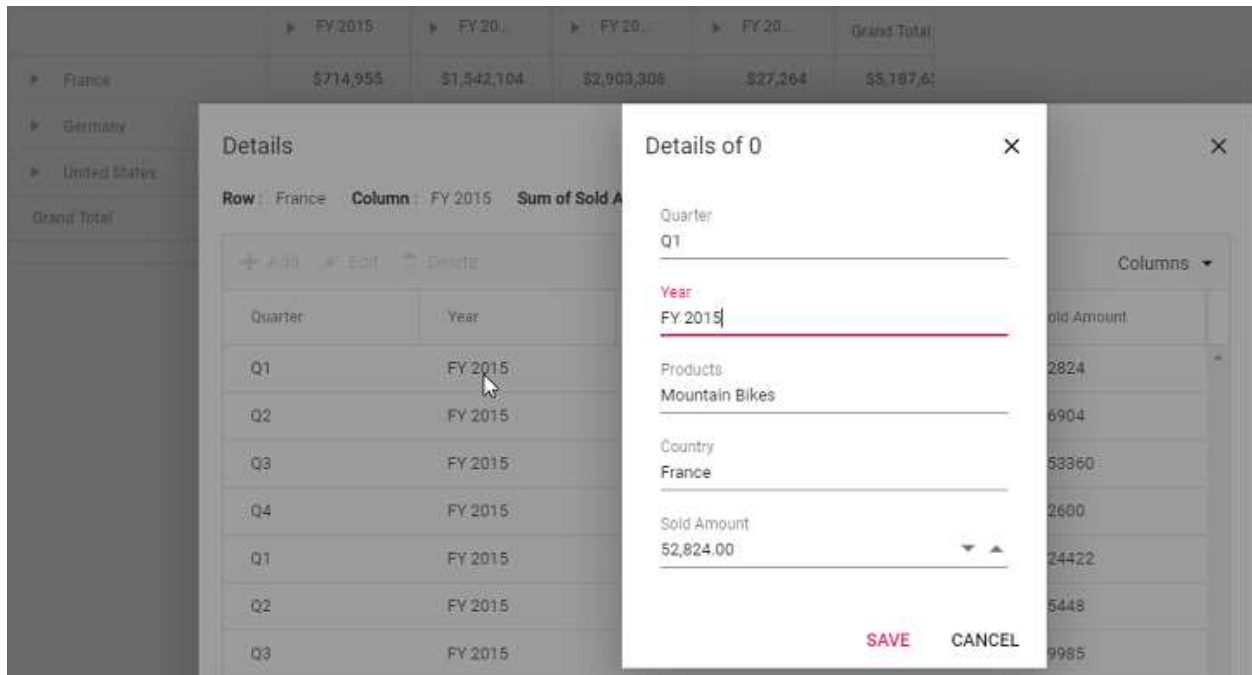
```

### DIALOG.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Batch

In batch edit mode, when you double-click the table cell, the state of target cell is changed to edit state.

You can perform bulk save (added, changed, and deleted data in the single request) to the data source by clicking the toolbar's **Update** button.

To enable the batch edit, set the [Mode](#) property in [PivotViewCellEditSettings](#) class to [EditMode.Batch](#).

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.PivotView.EditMode.Batch)).Render()
```

**BATCH.CS**

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

|               | FY 2015   | FY 20...    | FY 20...    | FY 20... | Grand Total |
|---------------|-----------|-------------|-------------|----------|-------------|
| France        | \$714,955 | \$1,542,104 | \$2,903,308 | \$27,264 | \$5,187,61  |
| Germany       |           |             |             |          |             |
| United States |           |             |             |          |             |
| Grand Total   |           |             |             |          |             |

Details

Row: France    Column: FY 2015    Sum of Sold Amount: \$714,955

+ Add    Delete    Update    Cancel

Columns

| Quarter | Year    | Products       | Country | Sold Amount |
|---------|---------|----------------|---------|-------------|
| Q1      | FY 2015 | Mountain Bikes | France  | 52824       |
| Q2      | FY 2015 | Mountain Bikes | France  | 86904       |
| Q3      | FY 2015 | Mountain Bikes | France  | 153360      |
| Q4      | FY 2015 | Mountain Bikes | France  | 42600       |
| Q1      | FY 2015 | Road Bikes     | France  | 124422      |
| Q2      | FY 2015 | Road Bikes     | France  | 85448       |

### Command column

An additional column appended in the grid layout holds the command buttons to perform the CRUD operation.

To enable the command columns, set the [AllowCommandColumns](#) property in [PivotViewCellEditSettings](#) class to **true**.

The available built-in command buttons are:

|                |                          |
|----------------|--------------------------|
| Command Button | Actions                  |
| -----          | -----                    |
| Edit           | Edit the current row.    |
| Delete         | Delete the current row.  |
| Save           | Update the edited row.   |
| Cancel         | Cancel the edited state. |

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSet
tings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
```

```

 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }) .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }) .EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowA
adding(true).AllowDeleting(true).AllowEditing(true).AllowCommandColumns(true)
).Render()

```

## CC.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015   | FY 20...    | FY 20...    | FY 20... | Grand Total |
|---------------|-----------|-------------|-------------|----------|-------------|
| France        | \$714,955 | \$1,542,104 | \$2,903,308 | \$27,264 | \$5,187,6   |
| Germany       |           |             |             |          |             |
| United States |           |             |             |          |             |
| Grand Total   |           |             |             |          |             |

Details

Row: France Column: FY 2015 Sum of Sold Amount: \$714,955

| Quarter | Year    | Products       | Country | Sold Amount | Manage Records |
|---------|---------|----------------|---------|-------------|----------------|
| Q1      | FY 2015 | Mountain Bi... | France  | 52824       |                |
| Q2      | FY 2015 | Mountain Bi... | France  | 86904       |                |
| Q3      | FY 2015 | Mountain Bi... | France  | 153360      |                |
| Q4      | FY 2015 | Mountain Bi... | France  | 42600       |                |
| Q1      | FY 2015 | Road Bikes     | France  | 124422      |                |
| Q2      | FY 2015 | Road Bikes     | France  | 85448       |                |

### Inline Editing

Allows editing of a value cell directly without the use of an external edit dialog. It is applicable if and only if a single raw data is used for the value of the cell. It is applicable to all editing modes, such as normal, batch, dialog and column commands. It can be enabled by setting the [allowInlineEditing](#) property in [editSettings](#) to **true**.

## CSHTML

```

@Html.EJS().PivotView("PivotGrid").Width("100%).Height("300").DataSourceSet
tings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>
)ViewBag.Data)
 .FormatSettings(formatsettings =>

```



```

{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Columns(columns =>
{
columns.Name("Date").Add(); columns.Name("Product").Add();
}).Rows(rows =>
{
rows.Name("Country").Add();
}).Values(values =>
{
values.Name("Quantity").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}
)).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowEditing(true).AllowInlineEditing(true)).Render()

```

### INLINE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                | FY 2005    |             |            |             |            |             | FY 2006    |             |
|----------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|
|                | Bike       |             | Car        |             | Van        |             | Bike       |             |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount |
| Canada         | 22         | \$2,100     | 26         | \$1,060     | 22         | \$3,100     | 40         | \$4,400     |
| France         | 49         | \$9,450     | 49         | \$4,800     | 43         | \$7,250     | 46         | \$5,280     |
| Germany        | 20         | \$5,900     | 45         | \$3,375     | 78         | \$3,400     | 58         | \$4,960     |
| United Kingdom | 47         | \$1,040     | 42         | \$1,450     | 53         | \$5,150     | 24         | \$2,400     |
| United States  | 47         | \$3,200     | 49         | \$3,320     | 45         | \$6,300     | 53         | \$4,680     |
| Grand Total    | 185        | \$21,690    | 211        | \$15,960    | 241        | \$25,200    | 221        | \$17,720    |

### Editing using the pivot chart

Users can also add, delete, or update the underlying raw items of any data point via pivot chart. The raw items will be shown in the data grid in the new window by clicking the appropriate data point. Then you can edit the raw items as mentioned above by any of the edit types (normal, dialog, batch and command column).

### CSHTML

```

@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{

```

```

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).AllowCommandColumns(true)).DisplayOption(new PivotViewDisplayOption { View = View.Chart }).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries => chartSeries.Type(ChartSeriesType.Column))).Render()

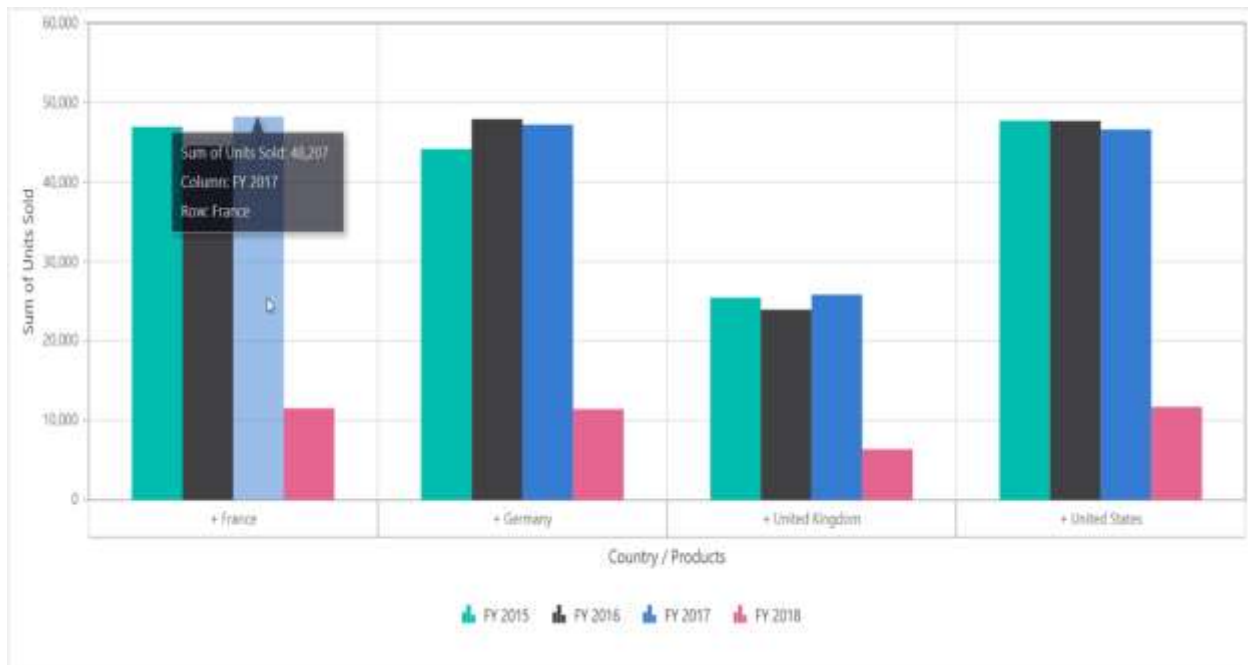
```

### CHART.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



<br/>

<br/>

### Events

#### EditCompleted

The event [editCompleted](#) triggers when values cells are edited completely. The event provides edited cell(s) information along with its previous cell value. It also helps to do the CRUD operation by manually updating the database which is connected to the component. It has the following parameters.

- **currentData** - It holds the current raw data of the edited cells.
- **previousData** - It holds the previous raw data of the edited cells.
- **previousPosition** - It holds the index of the raw data whose values are edited.
- **cancel** - It is a boolean property and if it is set as **true**, the editing won't be reflected in the pivot table.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
```

```

{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
}))).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowA
dding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.Pivot
View.EditMode.Normal)).EditCompleted("editCompleted").Render()
<script>
 function editCompleted(args) {
 //triggers when a value cell is edited.
 }
</script>

```

### EDITCOMPLETED.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### DrillThrough

For more information [refer](#) here.

### BeginDrillThrough

For more information [refer](#) here.

### ActionBegin

The event [actionBegin](#) triggers when the UI actions such as CRUD operations (via dialog) and inline editing begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

| Action  | Action Name         |
|---------|---------------------|
| -----   | -----               |
| Editing | Edit record         |
| Save    | Save edited records |
| Add     | Add new record      |
| Delete  | Remove record       |

- **cancel**: It allows user to restrict the current action.

In the below sample, editing actions such as add and save can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
 })
).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.PivotView.EditMode.Dialog)).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Add new record' || args.actionName == 'Save edited records') {
 args.cancel = true;
 }
 }
</script>
```

### ACTIONBEGIN-EDITING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionComplete

The event [actionComplete](#) triggers when the UI action such as CRUD operations (via dialog) or inline editing, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action | Action Name          |
|--------|----------------------|
| Save   | Edited records saved |
| Add    | New record added     |
| Delete | Record removed       |
| Update | Records updated      |

- **actionInfo**: It holds the unique information about the current UI action. For example, if save action is completed, the event argument contains information such as mode of editing and saved records.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
 })
).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.PivotView.EditMode.Dialog)).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'New record added' || args.actionName == 'Edited records saved') {
 // Triggers when the editing UI actions such as add and edit are completed.
 }
 }
}
```

```
</script>
```

### ACTIONCOMPLETE-EDITING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

| Action      | Action Name         |
|-------------|---------------------|
| ----- ----- |                     |
| Editing     | Edit record         |
| Save        | Save edited records |
| Add         | Add new record      |
| Delete      | Remove record       |

- **errorInfo**: It holds the error information of the current UI action.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1)
.UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}
```

```

 })).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowA
dding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.Pivot
View.EditMode.Dialog)).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Add new record' || args.actionName == 'Edit
record') {
 // Triggers when the current UI action fails to achieve the
desired result.
 }
 }
</script>

```

### ACTIONFAILURE-EDITING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

## Data Formatting

### Number Formatting

Allows you to specify the required display format that should be used in values of the pivot table. Supported display formats are:

- Number
- Currency
- Percentage
- Custom

You can apply format for the numeric values using the following properties in the [FormatSettings](#).

- [Name](#): It allows to specify the field name.
- [Format](#): It allows to specify the format of the respective field.

Possible formatting values are:

1. N - denotes numeric type.
2. C - denotes currency type.
3. P - denotes percentage type.

**Note:** If no format is specified it takes number as default format type.

Other properties include:

- [UseGrouping](#): It allows to enable or disable the separator, for example, \$100,000,000 or \$100000000 respectively. By default, it will be set as **true**.



- **Currency:** It allows to set the currency code which needs to be considered for the currency formatting.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C2").Currency("EUR").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(false).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .ShowFieldList(true).Render()
```

### FORMATTING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             |              | FY 2016    |             |
|---------------|------------|-------------|--------------|------------|-------------|
|               | Units Sold | Sold Amount | Total Amount | Units Sold | Sold Amount |
| France        | 450        | €714955.00  | 450          | 526        | €1542104    |
| Germany       | 440        | €563515.00  | 440          | 496        | €1772104    |
| United States | 546        | €754515.00  | 546          | 636        | €2263104    |
| Grand Total   | 1436       | €2032985.00 | 1436         | 1658       | €5577312    |

You can also format the values at runtime using the formatting dialog. This option can be enabled by setting the [AllowNumberFormatting](#) property to **true**. The same has been discussed in some of the upcoming topics.

### Custom format

You can add any custom format directly to the [Format](#) property in the [FormatSettings](#). Custom format can be achieved by using one or more format specifiers listed in the below table.

| Specifier                | Description                                                                                                                   | Input                            | Format Output |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------|----------------------------------|---------------|
| 0                        | Replaces the zero with the corresponding digit if it is present. Otherwise, zero will appear in the result string.            | { format: '0000' }               | '0123'        |
| #                        | Replaces the '#' symbol with the corresponding digit if it is present. Otherwise, no digits will appear in the result string. | { format: '####' }               | '1234'        |
| .                        | Denotes the number of digits permitted after the decimal point.                                                               | { format: '###0.##0#' }          | '546321.000'  |
| %                        | Percent specifier denotes the percentage type format.                                                                         | { format: '0000 %' }             | '0100 %'      |
| \$                       | Denotes the currency type format that is based on the global currency code specified.                                         | { format: '\$###.00' }           | '\$ 13.00'    |
| ;                        | Denotes separate formats for positive, negative and zero values.                                                              | { format: '###.##;(###.00);-0' } | '(120.00)'    |
| 'String' (single Quotes) | Denotes the characters that are enclosed in the single quote (') to be replaced in the resulting string.                      | { format: '####.00 '@' ' }       | '123.00 @'    |

**Note:** If custom format is defined, certain properties such as [UseGrouping](#) and [Currency](#) will not be considered.

### CSHTML

```
@{var amount = "\" + "Sum(Amount)" + "\"";}
@{var sold = "\" + "Sum(Sold)" + "\"";}
@{ var totalPrice = amount + "+" + sold;}
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Sold").Format("####.00
'Nos').MaximumSignificantDigits(10).MinimumSignificantDigits(1).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
values.Name("Total").Add();
}).CalculatedFieldSettings(calculatedfieldsettings =>
{
```

```

 calculatedfieldsettings.Name("Total").Formula(totalPrice).Add();
 }).AllowCalculatedField(true).ShowFieldList(true).Render()

```

## FORMATTING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015     |             |              | FY 2016     |             |              |
|---------------|-------------|-------------|--------------|-------------|-------------|--------------|
|               | Units Sold  | Sold Amount | Total Amount | Units Sold  | Sold Amount | Total Amount |
| France        | 450.00 Nos  | 714955      | 450          | 526.00 Nos  | 1542104     |              |
| Germany       | 440.00 Nos  | 563515      | 440          | 496.00 Nos  | 1772104     |              |
| United States | 546.00 Nos  | 754515      | 546          | 636.00 Nos  | 2263104     |              |
| Grand Total   | 1436.00 Nos | 2032985     | 1436         | 1658.00 Nos | 5577312     |              |

### Toolbar

You can enable formatting dialog option in the toolbar by adding **NumberFormatting** in the [Toolbar](#). After that, you can see the option to invoke the formatting dialog in the toolbar.

## CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings => {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add(); })
 .Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
 .Columns(columns => { columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add(); })
 .Values(values =>
 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product Categories").Add();
 })
 .GridSettings(new PivotViewGridSettings { ColumnWidth = 140 })
 .DisplayOption(new PivotViewDisplayOption { View = View.Both })
 .Toolbar(new List<string>
 () { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart", "Export", "SubTotal", "GrandTotal", "ConditionalFormatting",

```

```

"NumberFormatting" "FieldList"
}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport").RemoveReport("removeReport").NewReport("newReport").Render()
<style>
 #pivotview {
 width: 100%;
 height: 100%;
 }
 .e-tool-expand::before {
 content: '\e702';
 }
</style>
<script>
 function saveReport(args) {
 var reports = [];
 var isSaved = false;
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reports = JSON.parse(localStorage.pivotviewReports);
 }
 if (args.report && args.reportName && args.reportName != '') {
 reports.map(function (item) {
 if (args.reportName === item.reportName) {
 item.report = args.report;
 isSaved = true;
 }
 });
 if (!isSaved) {
 reports.push(args);
 }
 localStorage.pivotviewReports = JSON.stringify(reports);
 }
 }
 function fetchReport(args) {
 var reportCollection = [];
 var reeportList = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 reeportList.push(item.reportName);
 });
 args.reportName = reeportList;
 }
 function loadReport(args) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 args.report = item.report;
 }
 });
 }

```

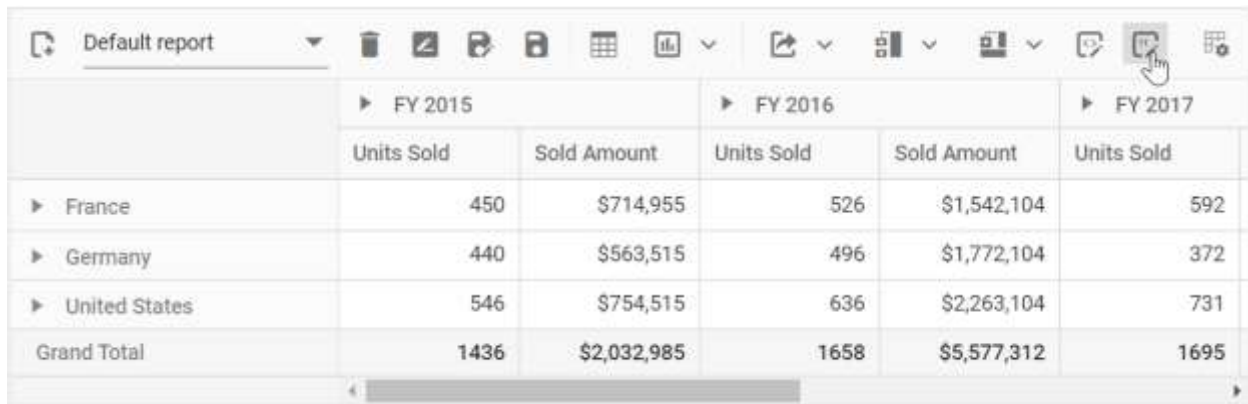
```

 }
 });
 if (args.report) {
 pivotObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
 }
}
function removeReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 for (var i = 0; i < reportCollection.length; i++) {
 if (reportCollection[i].reportName === args.reportName) {
 reportCollection.splice(i, 1);
 }
 }
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function renameReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 item.reportName = args.rename;
 }
 });
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function newReport() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.setProperties({
 dataSourceSettings: {
 columns: [],
 rows: [],
 values: [],
 filters: []
 }
 }, false);
}
</script>

```

**FORMATTING.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 592        |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 372        |
| United States | 546        | \$754,515   | 636        | \$2,263,104 | 731        |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1695       |

#### Invoking formatting dialog through external button

You can invoke the formatting dialog by clicking an external button using the `showNumberFormattingDialog` method.

#### CSHTML

```
@{var amount = "\" + "Sum(Amount)" + "\"; }
@{var sold = "\" + "Sum(Sold)" + "\"; }
@{ var totalPrice = amount + "+" + sold; }
@Html.EJS().Button("calculated-field-btn").Content("Calculated
Field").IsPrimary(true).Render()

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).CalculatedFieldSettings(calculatedfieldsettings =>
{
calculatedfieldsettings.Name("Total").Formula(totalPrice).Add();
```

```
 })).AllowCalculatedField(true).Render()
 <script>
 document.getElementById("calculated-field-
btn").addEventListener('click', function () {
 var pivotObj =
document.getElementById("PivotView").ej2_instances[0];
 pivotObj.calculatedFieldModule.createCalculatedFieldDialog();
 });
 </script>
```

### **FORMATTING.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

## Number Formatting

×

Values

All Values

Format Type

Number

Grouping

True

Decimal Places

0

Custom Format

APPLY

CANCEL

### Events

#### NumberFormatting

The event [NumberFormatting](#) fires while closing the number formatting dialog on "OK" button click. It allows the user to restrict the customization settings done by the user. It has the following parameters

- **formatName**: It holds the name of the field.
- **formatSettings**: It holds the [formatSettings](#) property of the pivot report.
- **cancel**: It is a boolean property and by setting this to true , the customization done in number formatting dialog won't be applied.



In the below sample, the customization done in number formatting dialog for the field "Amount" won't be applied.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolBar(true).AllowNumberFormatting(true).NumberFormatting("numberFormatting").ShowToolBar(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add(); })
.Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
.Columns(columns => { columns.Name("Year").Add();
columns.Name("Order_Source").Caption("Order Source").Add(); })
.Values(values =>
{
values.Name("In_Stock").Caption("In Stock").Add();
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.Filters(filters =>
{
filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).DisplayOption(new PivotViewDisplayOption { View = View.Both
}).ToolBar(new List<string>
() { "NumberFormatting" }).Render()
<style>
#pivotview {
width: 100%;
height: 100%;
}
.e-tool-expand::before {
content: '\e702';
}
</style>
<script>
function numberFormatting(args) {
if (args.formatName === 'Amount') {
args.cancel = true;
}
}
</script>
```

### FORMATTING.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

*See Also*

- [Customize number, date, and time values](#)
- [NumberFormatOptions](#)
- [Toolbar](#)

### Conditional Formatting

Allows end user to change the appearance of the pivot table value cells with its background color, font color, font family, and font size based on specific conditions.

The conditional formatting can be applied at runtime through the built-in dialog, invoked from the toolbar. To do so, set [AllowConditionalFormatting](#) and [ShowToolbar](#) properties in [PivotView](#) class to **true**. Also, include the item **ConditionalFormatting** within the [Toolbar](#) property in [PivotView](#) class. End user can now see the "Conditional Formatting" icon in toolbar UI automatically, which on clicking will invoke the formatting dialog to perform necessary operations.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").DataSourceSettings(
dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).
 EnableSorting(true)
 .DrilledMembers(drilledmembers =>
 {

drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product
Categories").Add();
 })
 .ConditionalFormatSettings(format =>
 {

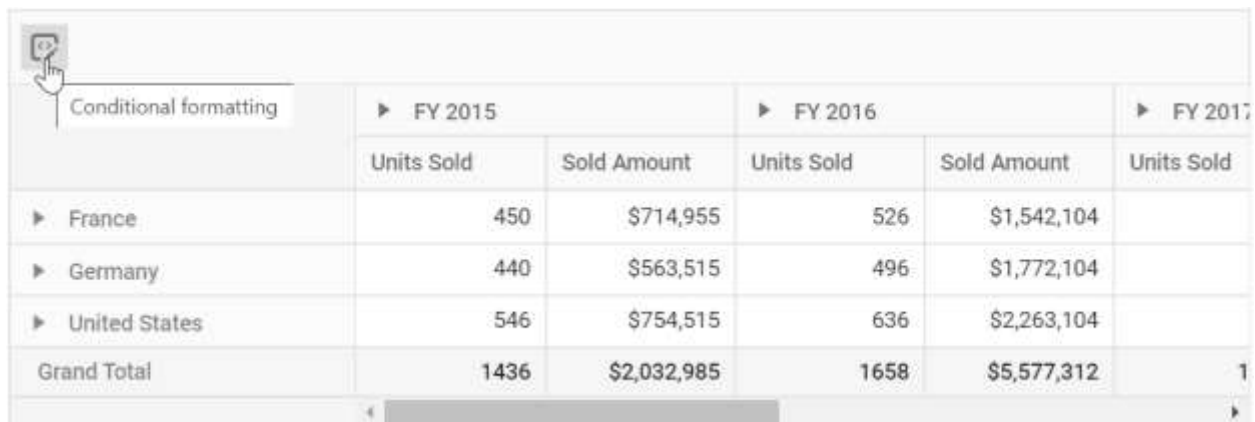
format.Conditions(Syncfusion.EJ2.PivotView.Condition.LessThan).Measure("In_Stock").Value1(1000).Style(style => {
style.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSize("12px"); }).Add();
```

```
format.Conditions(Syncfusion.EJ2.PivotView.Condition.Between).Measure("Sold")
).Value1(500).Value2(40000).Style(style => {
style.BackgroundColor("#f48fb1").Color("black").FontFamily("Tahoma").FontSiz
e("12px"); }).Add();
})
).AllowConditionalFormatting(true).ShowToolbar(true).Toolbar(new
List<string>() { "ConditionalFormatting" }).Render()
```

### FORMATTING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.drilledMembers = new string[] { "France" };
 return View();
}
```

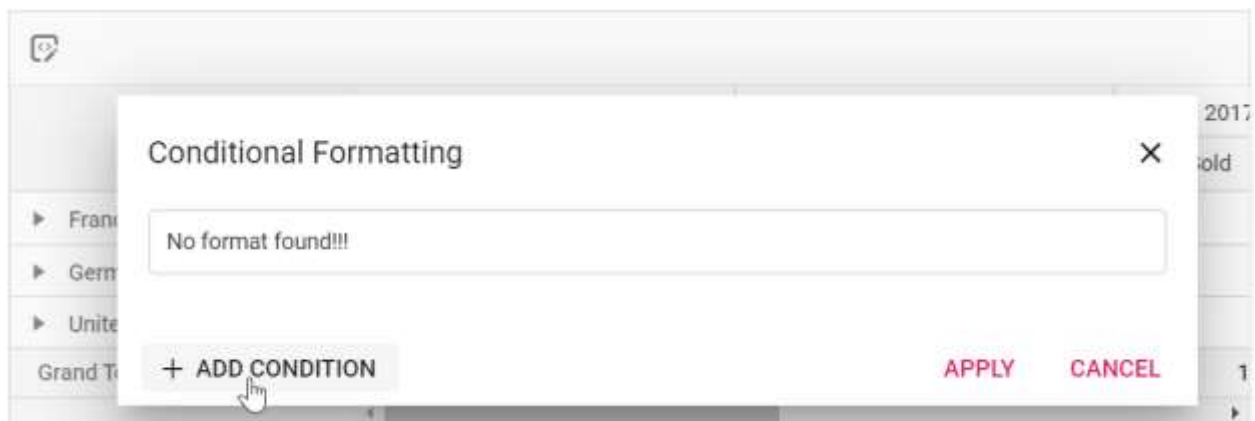
<!-- markdownlint-disable MD012 -->



|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

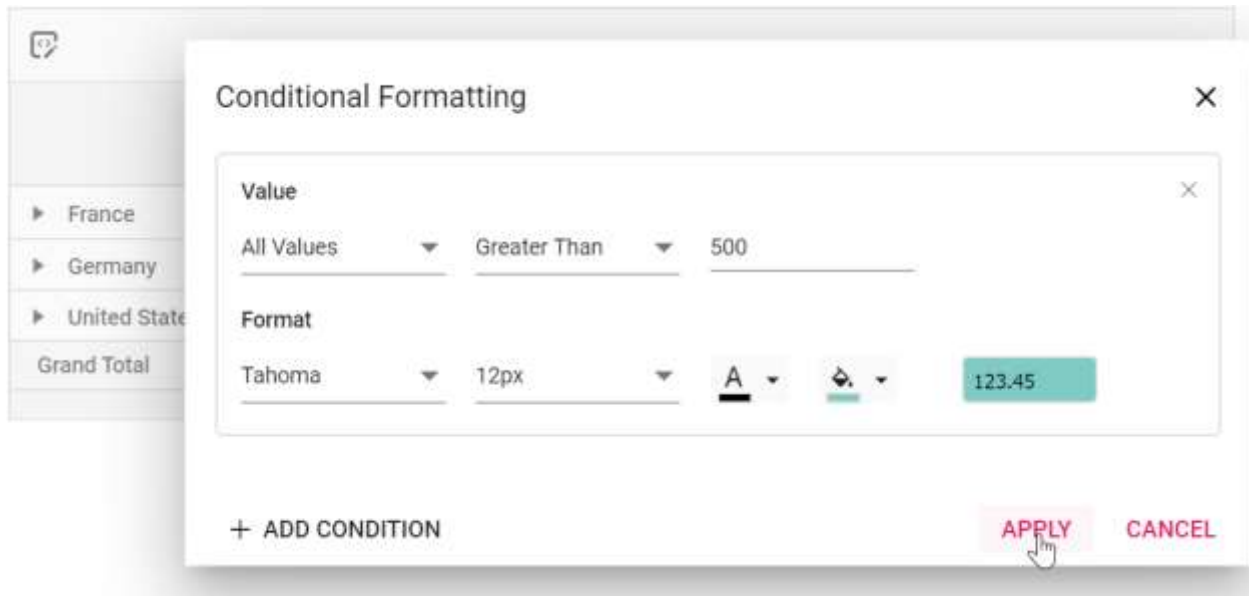
<br/>

<br/>



<br/>

&lt;br/&gt;



&lt;br/&gt;

&lt;br/&gt;

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |

Conditional formatting can also be included in the pivot table through code-behind using the [PivotViewConditionalFormatSetting](#) class. The required properties to apply a new conditional formatting are,

- [Measure](#): Specifies the value field name for which style will be applied.
- [Conditions](#): Specifies the operator type such as equals, greater than, less than, etc.
- [Value1](#): Specifies the start value.
- [Value2](#): Specifies the end value.
- [PivotViewStyle](#): Specifies the style for the cell.

The available style properties in [PivotViewStyle](#) class, to set in value cells are:

- [BackgroundColor](#): Specifies the background color.
- [Color](#): Specifies the font color.

- [FontFamily](#): Specifies the font family.
- [FontSize](#): Specifies the font size.

Meanwhile, user can also view conditional formatting dialog in UI by invoking `ShowConditionalFormattingDialog` method on an external button click which is shown in the below code sample.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("conditional-formatting-btn").Content("Conditional
Formatting").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSet
tings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .DrilledMembers(drilledmembers =>
 {
drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product
Categories").Add();
 })
 .ConditionalFormatSettings(format =>
 {
format.Conditions(Condition.LessThan).Measure("In_Stock").Value1(1000).Style
(style => {
style.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSiz
e("12px"); }).Add();

format.Conditions(Condition.Between).Measure("Sold").Value1(500).Value2(4000
0).Style(style => {
style.BackgroundColor("#f48fb1").Color("black").FontFamily("Tahoma").FontSiz
e("12px"); }).Add();
 })
).AllowConditionalFormatting(true).Render()
<script>
```

```

document.getElementById("conditional-formatting-
btn").addEventListener('click', function () {
 var pivotObj =
document.getElementById("PivotView").ej2_instances[0];

pivotObj.conditionalFormattingModule.showConditionalFormattingDialog();
});
</script>

```

### FORMATTING.CS

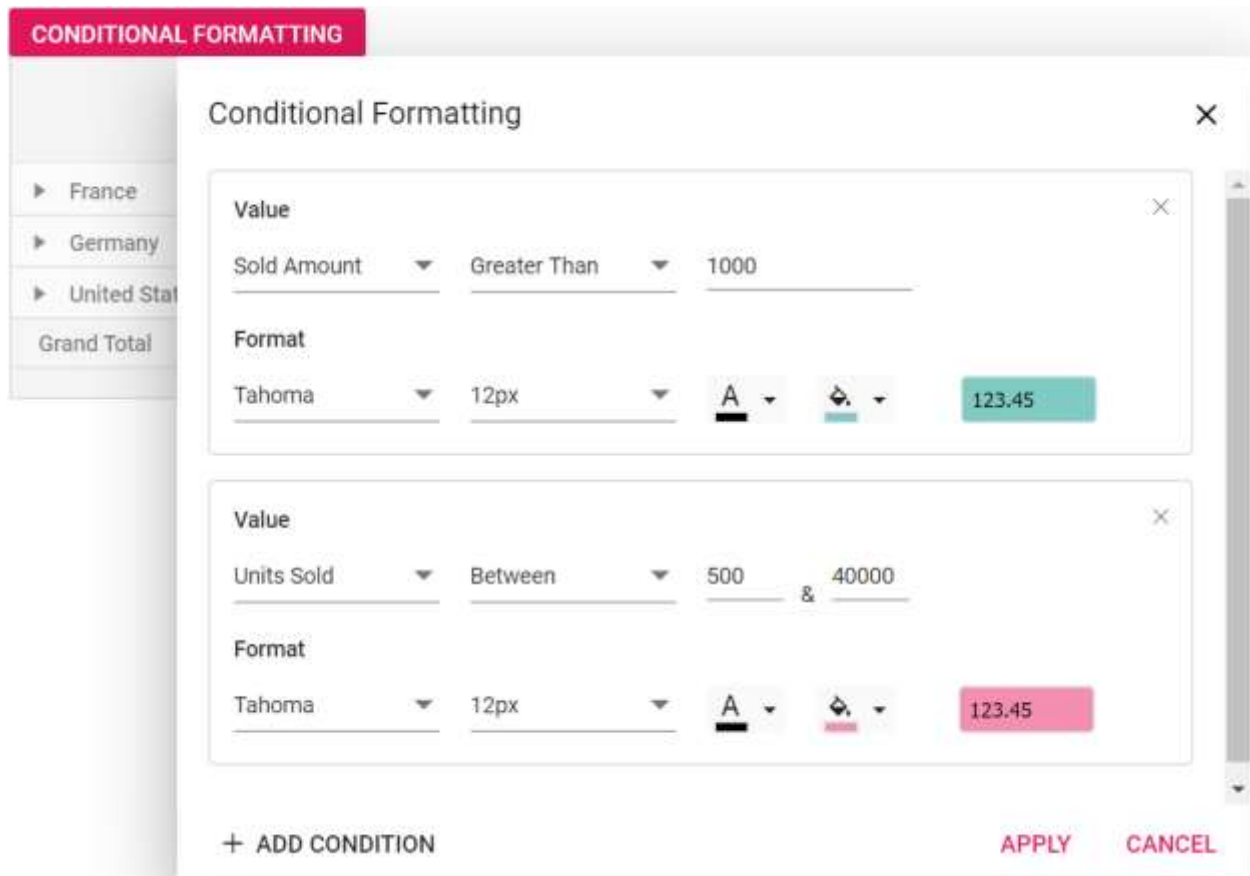
```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

**CONDITIONAL FORMATTING**

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |



#### *Conditional formatting for all fields*

Allows end user to apply conditional formatting commonly for all value fields just by ignoring the [Measure](#) property and setting rest of the properties in [PivotViewConditionalFormatSettings](#) class.

#### **CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .DrilledMembers(drilledmembers =>
 {
drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
```

```

 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product
Categories").Add();
 })
 .ConditionalFormatSettings(format =>
 {
 format.Conditions(Syncfusion.EJ2.PivotView.Condition.GreaterThan).Value1(500)
).Style(style => {
 style.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSiz
e("12px"); }).Add();
 })
).AllowConditionalFormatting(true).Render()

```

### FORMATTING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.drilledMembers = new string[] { "France" };
 return View();
}

```

|                 | ► FY 2015  |             | ► FY 2016  |             | ► FY 2017  |
|-----------------|------------|-------------|------------|-------------|------------|
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ► France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| ► Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| ► United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total     | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

#### Conditional formatting for specific value field

Allows end user to apply conditional formatting to a specific value field by setting the [Measure](#) property with specific value field name in [PivotViewConditionalFormatSettings](#) class.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").DataSourceSet
tings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
 .DrilledMembers(drilledmembers =>
 {
 drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();

```



```

 })
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product
Categories").Add();
 })
 .ConditionalFormatSettings(format =>
 {
 format.Conditions(Condition.LessThan).Measure("In_Stock").Value1(500).Style(
style => {
style.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSiz
e("12px"); }).Add();
 })
).AllowConditionalFormatting(true).Render()

```

## FORMATTING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |

### Conditional formatting for specific row or column

You can apply conditional formatting for specific row or column using [Label](#) option in the pivot table. It can be configured using the [PivotViewConditionalFormatSettings](#) tag through code behind, during initial rendering. The required settings are:

- [Label](#): Specifies the header name to apply conditions for row or column.
- [Conditions](#): Specifies the operator type such as equals, greater than, less than, etc.
- [Value1](#): Specifies the start value.
- [Value2](#): Specifies the end value.
- [PivotViewStyle](#): Specifies the style for the cell.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product Categories").Add();
 })
 .ConditionalFormatSettings(format =>
 {
 format.Conditions(Syncfusion.EJ2.PivotView.Condition.Between).Label("Germany")
 .Value1(500).Value2(50000).Style(style => {
 style.BackgroundColor("#f48fb1").Color("black").FontFamily("Tahoma").FontSize("12px");
 }).Add();
 })
).AllowConditionalFormatting(true).Render()
```

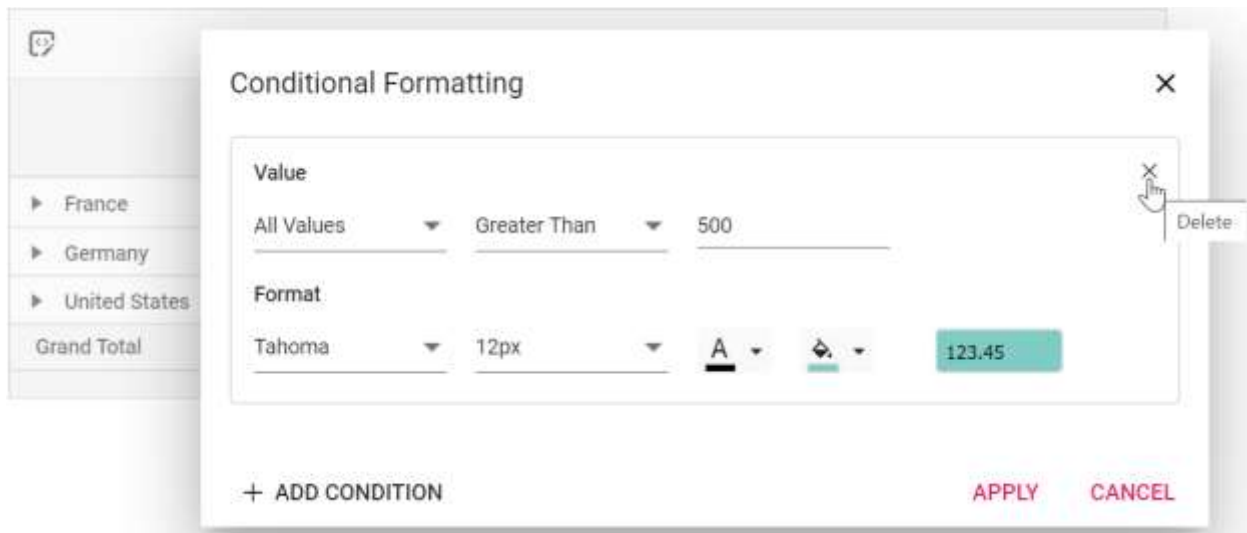
### LABELFORMATTING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
```

```
return View();
}
```

### Editing and removing existing conditional format

Editing and removing existing conditional format can be done through the UI at runtime. To do so, open the conditional formatting dialog and edit the "Value", "Condition" and "Format" options based on user requirement and click "OK". To remove a conditional format, click the "Delete" icon besides the respective condition.



### Event

#### ConditionalFormatting

The event **ConditionalFormatting** is triggered initially while clicking the "ADD CONDITION" button inside the conditional formatting dialog in-order to fill user specific condition instead of default condition at runtime. To use this event, **AllowConditionalFormatting** property in **PivotView** must be set to **true**. It has following parameters -

- **applyGrandTotals** - boolean property, by setting this to true user can enable formatting to grand totals.
- **conditions** - condition to be filled in conditional formatting dialog.
- **label** - Label value for conditional formatting dialog.
- **measure** - measure value for the conditional formatting dialog.
- **style** - style property of the conditional formatting dialog.
- **value1** - value 1 for conditional formatting dialog.
- **value2** - value 2 for conditional formatting dialog, this is applicable only for selected conditions like **Between** and **NotBetween**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("conditional-formatting-btn").Content("Conditional
Formatting").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSet
tings(dataSource =>
```

```

dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .DrilledMembers(drilledmembers =>
 {
drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("In_Stock").Caption("In Stock").Add();
 values.Name("Sold").Caption("Units Sold").Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product
Categories").Add();
 })
 .ConditionalFormatSettings(format =>
 {
format.Conditions(Condition.LessThan).Measure("In_Stock").Value1(1000).Style
(style => {
style.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSize("12px");
}).Add();
 })

).AllowConditionalFormatting(true).ConditionalFormatting("conditionalFormatting").Render()
<script>
 document.getElementById("conditional-formatting-
btn").addEventListener('click', function () {
 var pivotObj =
document.getElementById("PivotView").ej2_instances[0];

pivotObj.conditionalFormattingModule.showConditionalFormattingDialog();
 });
 function conditionalFormatting(args) {
 args.style.backgroundColor = "Blue";
 args.value1 = 23459;
 }
</script>

```

**FORMATTING.CS**

```

public ActionResult Index()
{

```

```

var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```

## Report Manipulation

### Grouping Bar

The Grouping Bar option in pivot table automatically populates fields from the bound data source and allows end users to drag fields between different axes such as columns, rows, values, and filters, and create pivot table at runtime. It can be enabled by setting the [showGroupingBar](#) property in [ejs-pivotview](#) tag to **true**.

Similar to Field List, Grouping Bar UI also comes with basic interactions like,

- Re-arranging fields through drag-and-drop operation between row, column, value and filter axes.
- Remove fields from the existing report using remove icon.
- Add fields to the report using fields panel option.
- Filtering members of specific fields using filter icon.
- Sorting members of specific fields using sort icon.

**Note:** The grouping bar provides some additional options to customize it's UI using [ejs-groupingBarSettings](#) property.

### CSHTML

```

@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)Model).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).Render()

```

### GROUPINGBAR.CS

```

public ActionResult Index()
{

```

```

var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Sold = 31, Amount = 52824, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 27, Amount = 46008, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 127800, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 69, Amount = 117576, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country =
"France", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 57, Amount = 85448, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 67, Amount = 70008, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 89, Amount = 60496, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 801880, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 57, Amount = 204168, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 75, Amount = 737800, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 87, Amount = 884168, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 39, Amount = 729576, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 38860, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
 pivotData.Add(new PivotData { Sold = 93, Amount = 139412, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });

```

```

pivotData.Add(new PivotData { Sold = 51, Amount = 92824, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 61, Amount = 76904, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 43360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 85, Amount = 62600, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 97, Amount = 86008, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 69, Amount = 93496, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 49, Amount = 89576, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 40, Amount = 33360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 96, Amount = 77264, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 23, Amount = 24422, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 67, Amount = 75448, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 52345, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 13, Amount = 135612, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 57, Amount = 90008, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 29, Amount = 90496, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 99, Amount = 829576, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 80, Amount = 38360, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 91, Amount = 67824, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 81, Amount = 99904, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q2" });

```

```

pivotData.Add(new PivotData { Sold = 70, Amount = 49360, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 65, Amount = 69600, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 57, Amount = 90008, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 29, Amount = 90496, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q2" });
pivotData.Add(new PivotData { Sold = 85, Amount = 391880, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 97, Amount = 904168, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 85, Amount = 237800, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 77, Amount = 384168, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q2" });
pivotData.Add(new PivotData { Sold = 99, Amount = 829576, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 80, Amount = 38360, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 76, Amount = 97264, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2018", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 53, Amount = 94422, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"
});
pivotData.Add(new PivotData { Sold = 90, Amount = 45448, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"
});
pivotData.Add(new PivotData { Sold = 29, Amount = 92345, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"
});
pivotData.Add(new PivotData { Sold = 67, Amount = 235612, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"
});
pivotData.Add(new PivotData { Sold = 97, Amount = 90008, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"
});
pivotData.Add(new PivotData { Sold = 79, Amount = 90496, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"
});
pivotData.Add(new PivotData { Sold = 95, Amount = 501880, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"
});
pivotData.Add(new PivotData { Sold = 97, Amount = 104168, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});

```



```

 pivotData.Add(new PivotData { Sold = 95, Amount = 837800, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 87, Amount = 684168, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 109, Amount = 29576, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});
 return pivotData;
 }
}
public class PivotData
{
 public int Sold { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string Products { get; set; }
 public string Year { get; set; }
 public string Quarter { get; set; }
}

```

|                    |                      |             |            |             |            |
|--------------------|----------------------|-------------|------------|-------------|------------|
| Sum of Units Sold  | Drop filter here     |             |            |             |            |
| Sum of Sold Amount | Year ↑ ⇅ Quarter ↑ ⇅ |             |            |             |            |
| Country ↑ ⇅        | FY 2015              |             | FY 2016    |             |            |
| Products ↑ ⇅       | Units Sold           | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▶ France           | 450                  | \$714,955   | 526        | \$1,542,104 |            |
| ▶ Germany          | 440                  | \$563,515   | 496        | \$1,772,104 |            |
| ▶ United States    | 546                  | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total        | 1436                 | \$2,032,985 | 1658       | \$5,577,312 |            |

### Show or hide fields panel

The fields panel, which is positioned above the grouping bar, displays the fields that are available in the data source but are not bound in the report. The fields can be dragged and dropped into the appropriate axis. In addition, any field removed from any axes will be automatically added to the fields panel. The fields panel can be displayed by setting the [showFieldsPanel](#) property in the [e-groupingBarSettings](#) to **true**.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
new PivotViewGroupingBarSettings {
ShowFieldsPanel = false }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}
)
)

```

```

 }).Rows(rows =>
 {
 rows.Name("Country").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 })).ShowGroupingBar(true).Render()

```

### SHOWFIELDSPANEL.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

| Country       | FY 2015 | FY 2016 | FY 2017 | FY 2018 | Grand Total |
|---------------|---------|---------|---------|---------|-------------|
| France        | 450     | 526     | 592     | 16      | 1584        |
| Germany       | 440     | 496     | 372     | 96      | 1404        |
| United States | 546     | 636     | 731     | 76      | 1989        |
| Grand Total   | 1436    | 1658    | 1695    | 188     | 4977        |

#### Show or hide all filter icon

The Grouping Bar has an option to filter members of particular fields at runtime in pivot table. In-order to filter members in a field, click the filter icon and check/uncheck members that needs to be displayed. By default, filter icon besides each field is enabled in the grouping bar. To disable the filter icon, set the property [showFilterIcon](#) in [e-groupingBarSettings](#) tag to **false**.

**Note:** By default, the filter icon is enabled in the grouping bar.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 ShowFilterIcon = false
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>

```

```
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).Render()
```

### SHOWFILTER.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                    |                  |             |            |             |            |
|--------------------|------------------|-------------|------------|-------------|------------|
| Sum of Units Sold  | Drop filter here |             |            |             |            |
| Sum of Sold Amount | Year Quarter     |             |            |             |            |
| Country            | FY 2015          |             | FY 2016    |             |            |
| Products           | Units Sold       | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France             | 450              | \$714,955   | 526        | \$1,542,104 |            |
| Germany            | 440              | \$563,515   | 496        | \$1,772,104 |            |
| United States      | 546              | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total        | 1436             | \$2,032,985 | 1658       | \$5,577,312 |            |

*Show or hide specific filter icon*

To disable the filter icon for a specific field, set the property [showFilterIcon](#) to **false** to the corresponding field in [e-datasourcesettings](#).

In the below sample, the filter icon of "Quarter" and "Products" fields have been hidden.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").DataSourceSettings(
dataSource =>
{
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add();
rows.Name("Products").ShowFilterIcon(false).Add();
}).Columns(columns =>
{

```

```

columns.Name("Year").Add();
columns.Name("Quarter").ShowFilterIcon(false).Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowGroupingBar(true).Render()

```

### SHOWFILTER.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                    |                      |             |            |             |            |
|--------------------|----------------------|-------------|------------|-------------|------------|
| Sum of Units Sold  | Drop filter here     |             |            |             |            |
| Sum of Sold Amount | Year ↑ ⇅ Quarter ↑ ⇅ |             |            |             |            |
| Country ↑ ⇅        | FY 2015              |             | FY 2016    |             |            |
| Products ↑         | Units Sold           | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France             | 450                  | \$714,955   | 526        | \$1,542,104 |            |
| Germany            | 440                  | \$563,515   | 496        | \$1,772,104 |            |
| United States      | 546                  | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total        | 1436                 | \$2,032,985 | 1658       | \$5,577,312 |            |

#### Show or hide all sort icon

The Grouping Bar has an option to order members of a particular fields either in ascending or descending at runtime. In order to sort a field, click the sort icon and to reverse its sort direction, once again click the same sort icon. By default, the sort icon besides each field is enabled in the grouping bar and members will be arranged in ascending order. To disable the sort option, set the property [showSortIcon](#) in [e-groupingBarSettings](#) tag to **false**.

**Note:** By default, the sort icon is enabled in the grouping bar.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 ShowSortIcon = false
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>

```

```
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowGroupingBar(true).Render()
```

### SHOWSORT.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                    |                  |             |            |             |       |
|--------------------|------------------|-------------|------------|-------------|-------|
| Sum of Units Sold  | Drop filter here |             |            |             |       |
| Sum of Sold Amount | Year Quarter     |             |            |             |       |
| Country            | FY 2015          |             | FY 2016    |             |       |
| Products           | Units Sold       | Sold Amount | Units Sold | Sold Amount | Units |
| France             | 450              | \$714,955   | 526        | \$1,542,104 |       |
| Germany            | 440              | \$563,515   | 496        | \$1,772,104 |       |
| United States      | 546              | \$754,515   | 636        | \$2,263,104 |       |
| Grand Total        | 1436             | \$2,032,985 | 1658       | \$5,577,312 |       |

#### Show or hide specific sort icon

To disable the sort icon for a specific button, set the property [showSortIcon](#) to **false** to the corresponding field in [e-datasourcesettings](#).

In the below sample, the sort icon of "Quarter" and "Country" fields have been hidden.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
dataSource =>
{
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").ShowSortIcon(false).Add(); rows.Name("Products").Add();
```

```

}).Columns(columns =>
{
columns.Name("Year").Add();
columns.Name("Quarter").ShowSortIcon(false).Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowGroupingBar(true).Render()

```

### SHOWSORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                    |                  |             |            |             |            |
|--------------------|------------------|-------------|------------|-------------|------------|
| Sum of Units Sold  | Drop filter here |             |            |             |            |
| Sum of Sold Amount | Year ↑ Quarter   |             |            |             |            |
| Country            | FY 2015          |             | FY 2016    |             |            |
| Products           | Units Sold       | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France             | 450              | \$714,955   | 526        | \$1,542,104 |            |
| Germany            | 440              | \$563,515   | 496        | \$1,772,104 |            |
| United States      | 546              | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total        | 1436             | \$2,032,985 | 1658       | \$5,577,312 |            |

*Show or hide all remove icon*

The Grouping Bar has an option to remove any field at runtime. To remove a field, just click the remove icon. By default, the remove icon besides each field is enabled in the grouping bar. To disable the remove icon, set the property [showRemoveIcon](#) in [e-groupingBarSettings](#) tag to **false**.

**Note:** By default, the remove icon is enabled in the grouping bar.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 ShowRemoveIcon = false
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10)
 .MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {

```

```
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .ShowGroupingBar(true).Render()
```

### SHOWREMOVE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                      |                      |             |            |             |            |
|----------------------|----------------------|-------------|------------|-------------|------------|
| Sum of Units Sold ▾  | Drop filter here     |             |            |             |            |
| Sum of Sold Amount ▾ | Year ↑ ▾ Quarter ↑ ▾ |             |            |             |            |
| Country ↑ ▾          | FY 2015              |             | FY 2016    |             |            |
| Products ↑ ▾         | Units Sold           | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▶ France             | 450                  | \$714,955   | 526        | \$1,542,104 |            |
| ▶ Germany            | 440                  | \$563,515   | 496        | \$1,772,104 |            |
| ▶ United States      | 546                  | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total          | 1436                 | \$2,032,985 | 1658       | \$5,577,312 |            |

*Show or hide specific remove icon*

To disable the remove icon for a specific button, set the property [showRemoveIcon](#) to **false** to the corresponding field in [e-datasourcesettings](#).

In the below sample, the remove icon of fields "Year", "Sold" and "Products" have been hidden.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
dataSource =>
{
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).
EnableSorting(true).FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).
MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add();
rows.Name("Products").ShowRemoveIcon(false).Add();
}
```

```

 }).Columns(columns =>
 {
 columns.Name("Year").ShowRemoveIcon(false).Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").ShowRemoveIcon(false).Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).Render()

```

### SHOWREMOVE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                        |                        |             |            |             |     |
|------------------------|------------------------|-------------|------------|-------------|-----|
| Sum of Units Sold ▾    | Drop filter here       |             |            |             |     |
| Sum of Sold Amount ▾ ⊗ | Year ↑ ▾ Quarter ↑ ▾ ⊗ |             |            |             |     |
| Country ↑ ▾ ⊗          | FY 2015                |             | FY 2016    |             |     |
| Products ↑ ▾           | Units Sold             | Sold Amount | Units Sold | Sold Amount | Uni |
| ▶ France               | 450                    | \$714,955   | 526        | \$1,542,104 |     |
| ▶ Germany              | 440                    | \$563,515   | 496        | \$1,772,104 |     |
| ▶ United States        | 546                    | \$754,515   | 636        | \$2,263,104 |     |
| Grand Total            | 1436                   | \$2,032,985 | 1658       | \$5,577,312 |     |

#### Disable all fields from dragging

The Grouping Bar has an option to drag-and-drop fields between row, column, value and filter axes in order to change report at runtime. By default, all fields are available for drag-and-drop operation in the grouping bar. To disable these fields, set the property [allowDragAndDrop](#) in [e-groupingBarSettings](#) tag to **false**. This will prevent end user from changing the current report.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
 AllowDragAndDrop =false }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{

```



```
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .ShowGroupingBar(true).Render()
```

**DRAG.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             |            |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |            |

*Disable specific field from dragging*

To disable dragging for a specific button, set the property [allowDragAndDrop](#) to **false** to the corresponding field in [e-datasourcesettings](#).

In the below sample, the drag and drop of the fields "Year" and "Products" have been restricted.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).FormatSettings(
 formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 })
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").AllowDragAndDrop(false).Add();
 })
```

```

 }).Columns(columns =>
 {
 columns.Name("Year").AllowDragAndDrop(false).Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).Render()

```

### **DRAG.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### *Remove specific field(s) from displaying*

When a report is bound to the pivot table, fields will be automatically populated within the Grouping Bar. In this case, you can also prevent specific fields from being displayed. To do so, set the appropriate field name(s) in the [excludeFields](#) property of [dataSourceSettings](#).

**Note:** The `excludeFields` property setting will be reflected in the field list UI as well, and for more information, see this [link](#).

### **CSHTML**

```

@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ExcludeFields(ViewBag.Excludefields)).ShowGroupingBar(true).Render()

```

### **REMOVESPECIFIC.CS**

```

public ActionResult Index()
{

```

```

var data = GetPivotData();
ViewBag.DataSource = data;
ViewBag.Excludefields = new string[] { "Amount" };
return View();
}

```

| Sum of Units Sold | Drop filter here |         |         |         |
|-------------------|------------------|---------|---------|---------|
|                   | Year             | Quarter |         |         |
| Country           | FY 2015          | FY 2016 | FY 2017 | FY 2018 |
| Products          |                  |         |         |         |
| France            | 450              | 526     | 592     |         |
| Germany           | 440              | 496     | 372     |         |
| United States     | 546              | 636     | 632     |         |
| Grand Total       | 1436             | 1658    | 1596    |         |

#### Changing aggregation type of value fields at runtime

End user can perform calculations over a group of values using the aggregation option. The value fields bound to the field list, appears with a dropdown icon, helps to select an appropriate aggregation type at runtime. On selection, the values in the Pivot Table will be changed dynamically. By default, the icon to set aggregation type is enabled in the grouping bar. To disable this icon, set the property [showValueTypeIcon](#) in [e-groupingBarSettings](#) tag to **false**. To know more about aggregation, [refer](#) here.

#### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = false
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).Render()

```

**AGGREGATION.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

<!-- markdownlint-disable MD012 -->

|                    |            |             |
|--------------------|------------|-------------|
| Sum of Units Sold  | Year       | Quarter     |
| Sum of Sold Amount | FY 2015    | FY 2016     |
| Country            | Units Sold | Sold Amount |
| Products           | Units Sold | Sold Amount |

<br/>

<br/>

<br/>

|                    |                  |
|--------------------|------------------|
| Sum of Units Sold  | Drop filter here |
| Sum of Sold Amount | Year             |
| Country            | Quarter          |
| Products           | FY 2015          |
| France             | Units Sold       |
| Germany            | Sold Amount      |
| United States      | Units Sold       |
| Grand Total        | 1436             |

*Show or hide specific dropdown icon*

To disable the dropdown icon for a specific button, set the property [showValueTypeIcon](#) to **false** to the corresponding field in [e-datasourcesettings](#).

In the below sample, the dropdown icon of field "Sold" is hidden.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
```

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true).FormatSettings(
 formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").ShowValueTypeIcon(false).Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
).ShowGroupingBar(true).Render()
```

### AGGREGATION-SPECIFIC.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                        |                          |             |            |             |            |
|------------------------|--------------------------|-------------|------------|-------------|------------|
| Sum of Units Sold ✕    | Drop filter here         |             |            |             |            |
| Sum of Sold Amount ▼ ✕ | Year ↑ = ✕ Quarter ↑ = ✕ |             |            |             |            |
| Country ↑ = ✕          | FY 2015                  |             | FY 2016    |             |            |
| Products ↑ = ✕         | Units Sold               | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France                 | 450                      | \$714,955   | 526        | \$1,542,104 |            |
| Germany                | 440                      | \$563,515   | 496        | \$1,772,104 |            |
| United States          | 546                      | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total            | 1436                     | \$2,032,985 | 1658       | \$5,577,312 |            |

**Note:** The property [showFilterIcon](#), [showSortIcon](#), [showValueTypeIcon](#) and [allowDragAndDrop](#) in fields of [e-datasourcesettings](#) are applicable for both grouping bar and field list.

#### Show values button

During runtime, the **Values** button in the grouping bar can be moved to a different position (i.e., different index) among other fields in the column or row axis. To enable the **Values** button, set the [showValuesButton](#) property to **true**.

**Note:** This support is only available for relational data sources.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Width("100%").ShowGroupingBar(true).ShowValuesButton(true).Height("350").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
)
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Add();
 })
).Render()
```

**GROUPINGBAR-VALUESBUTTON.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                    |                  |             |            |             |
|--------------------|------------------|-------------|------------|-------------|
| Sum of Units Sold  | Drop filter here |             |            |             |
| Sum of Sold Amount | Year             |             |            |             |
| Country            | Quarter          |             |            |             |
| Products           | Values           |             |            |             |
|                    | FY 2015          |             |            |             |
|                    | Q1               |             | Q2         |             |
|                    | Units Sold       | Sold Amount | Units Sold | Sold Amount |
| France             | 114              | 177246      | 108        |             |
| Mountain Bikes     | 31               | 52824       | 51         |             |
| Road Bikes         | 83               | 124422      | 57         |             |
| Germany            | 74               | 117246      | 128        |             |

*Event*[OnFieldDropped](#)

The event [onFieldDropped](#) fires on whenever a field is dropped over an axis.

It has following parameters - `droppedAxis`, `droppedField` and `dataSourceSettings`. In this sample we have modified the `droppedField` caption based on the `droppedAxis`.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).OnFieldDropped("onFieldDropped").Render()
<script>
 function onFieldDropped(args) {
 args.droppedField.caption = args.droppedField.name + " --> " +
args.droppedAxis;
 }
</script>
```

### FIELD-DROPPED.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### FieldDragStart

The event [fieldDragStart](#) fires whenever a field drag starts from its axis. It allows the user to restrict the drag operation based on its parameters. It has the following parameters

- `fieldName`: It holds the name of the appropriate field.
- `fieldItem`: It holds the complete definition of the appropriate field mentioned in data source settings.
- `axis`: It holds the axis name where the draggable field lies.
- `cancel`: It is a boolean property and by setting this to true, user can restrict the field from dragging.

In the below sample, the drag operation for the fields in row axis alone is restricted.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).FieldDragStart("fieldDragStart").Render()
<script>
 function fieldDragStart(args) {
 if(args.axis === 'rows') {
 args.cancel = true;
 }
 }
</script>
```

**FIELD-DRAG-START.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**FieldDrop**

The event [fieldDrop](#) fires whenever a field is dropped into an axis. It allows the user to restrict the drop operation based on its parameters. It has the following parameters

- **fieldName**: It holds the name of the appropriate field.
- **dropField**: It holds the complete definition of the appropriate field mentioned in data source settings.
- **draggedAxis**: It holds the axis name from where dragging was started.
- **dropAxis**: It holds the axis name where the field is dropped.
- **dropPosition**: It holds the dropped index among other existing fields in the axis.
- [dataSourceSettings](#): It holds complete pivot report.
- **cancel**: It is a boolean property and by setting this to true, user can restrict the field from being dropped.



In the below sample, dropping of any fields in value axis alone is restricted.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).FieldDrop("fieldDrop").Render()
<script>
 function fieldDrop(args) {
 if(args.dropAxis === 'values') {
 args.cancel = true;
 }
 }
</script>
```

### FIELD-DROP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### FieldRemove

The event [fieldRemove](#) fires when removing any field from their axis. It helps the user to limit the elimination of a field based on its parameters. It has the following parameters

- **fieldName**: It holds the name of the field to be removed.
- **fieldItem**: It holds the complete definition of the appropriate field mentioned in data source settings.
- **axis**: It holds the name of the axis from where it is to remove the field.
- **dataSourceSettings**: It holds complete pivot report.
- **cancel**: It is a boolean property and by setting this to true, user can restrict the field from removing.

In the below sample, the field "Country" could not be removed from report by any UI operations.

### **CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).FieldRemove("fieldRemove").Render()
<script>
 function fieldRemove(args) {
 if(args.fieldName === 'Country') {
 args.cancel = true;
 }
 }
</script>
```

### **FIELD-REMOVE.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### [AggregateMenuOpen](#)

The event [aggregateMenuOpen](#) fires while clicking dropdown icon of the value field button UI. It allows to customize the aggregate types to be displayed in the dropdown menu. It has the following parameters

- **fieldName**: It holds the name of the field that opens the aggregate menu.
- **aggregateTypes**: It holds the aggregation types set for a field.
- **displayMenuCount**: It allows to set the menu count to be displayed initially. By default, its count is 7.
- **cancel**: It is a boolean property and by setting this to true, dropdown menu won't be displayed.

In the below sample, the aggregate types of the field "Amount" has been customized in it's dropdown menu.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).AggregateMenuOpen("aggregateMenuOpen").Render()
<script>
 function aggregateMenuOpen(args) {
 args.displayMenuCount = 4;
 if(args.fieldName === 'Amount') {
 args.aggregateTypes = ['Sum', 'Avg', 'Max'];
 }
 }
</script>
```

### AGGREGATION-MENU-OPEN.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

| Drop filter here   |                | Year |             | Quarter |             |
|--------------------|----------------|------|-------------|---------|-------------|
|                    |                |      |             |         |             |
| Sum of Units Sold  | Sum            |      |             |         |             |
| Sum of Sold Amount | Count          |      |             |         |             |
| Country            | Distinct Count |      |             |         |             |
| Products           | Product        |      |             |         |             |
|                    | More...        |      |             |         |             |
| France             |                | 450  | \$714,955   | 526     | \$1,542,104 |
| Germany            |                | 440  | \$563,515   | 496     | \$1,772,104 |
| United States      |                | 546  | \$754,515   | 636     | \$2,263,104 |
| Grand Total        |                | 1436 | \$2,032,985 | 1658    | \$5,577,312 |

**Note:** The events [aggregateMenuOpen](#), [fieldRemove](#), [fieldDrop](#), [fieldDragStart](#) and [onFieldDropped](#) are applicable for both grouping bar and field list.

#### ActionBegin

The event [actionBegin](#) triggers when the UI action such as sorting, filtering, aggregation or edit calculated field, that are present in the grouping bar UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings:** It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName:** It holds the name of the current action began. The following are the UI actions and their names:

| Action                                      | Action Name           |
|---------------------------------------------|-----------------------|
| Sort icon                                   | Sort field            |
| Filter icon                                 | Filter field          |
| Aggregation (Value type drop down and menu) | Aggregate field       |
| Remove icon                                 | Remove field          |
| Edit icon                                   | Edit calculated field |

- **fieldInfo:** It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel:** It allows user to restrict the current action.

In the below sample, grouping bar UI actions such as sorting and filtering can be restricted by setting the **args.cancel** option to **true** in the [actionBegin](#) event.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 AllowDragAndDrop = true
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
```

```

values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
))).ShowGroupingBar(true).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Sort field' || args.actionName == 'Filter
field') {
 args.cancel = true;
 }
 }
</script>

```

### ACTIONBEGIN-AGGREGATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### ActionComplete

The event [actionComplete](#) triggers when the UI action such as as sorting, filtering, aggregation or edit calculated field, that are present in the grouping bar UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action                                      | Action Name             |
|---------------------------------------------|-------------------------|
| -----                                       | -----                   |
| Sort icon                                   | Field sorted            |
| Filter icon                                 | Field filtered          |
| Aggregation (Value type drop down and menu) | Field aggregated        |
| Remove icon                                 | Field removed           |
| Edit icon                                   | Calculated field edited |

- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **actionInfo**: It holds the unique information about the current UI action. For example, if sorting is completed, the event argument contains information such as sort order and the field name.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(new PivotViewGroupingBarSettings {
 AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'Field sorted' || args.actionName == 'Field
 filtered') {
 // Triggers when the grouping bar UI actions such as sorting and
 filtering are completed.
 }
 }
</script>
```

**ACTIONCOMPLETE-AGGREGATION.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**ActionFailure**

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- actionName:** It holds the name of the current action failed. The following are the UI actions and their names:

| Action | Action Name|

|-----|-----|

| Sort icon| Sort field|

| Filter icon| Filter field|

| Aggregation (Value type drop down and menu)| Aggregate field|

| Remove icon| Remove field|  
| Edit icon| Edit calculated field|

- **errorInfo**: It holds the error information of the current UI action.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
AllowDragAndDrop =true }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowGroupingBar(true).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Sort field' || args.actionName == 'Filter
field') {
 // Triggers when the current UI action fails to achieve the
desired result.
 }
 }
}
</script>
```

### ACTIONFAILURE-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*See Also*

- [Change load limited data in member editor](#)
- [Customize the icons for pivot table](#)

### Pivot Field List in ASP.NET MVC Pivot Table Component

The pivot table provides a built-in Field List similar to Microsoft Excel. It allows to add or remove fields and also rearrange them between different axes, including column, row, value, and filter along with sort and filter options dynamically at runtime.

The field list can be displayed in two different formats to interact with pivot table. They are:

- **In-built Field List (Popup):** To display the field list icon in pivot table UI to invoke the built-in dialog.
- **Stand-alone Field List (Fixed):** To display the field list in a static position within a web page.

#### *In-built Field List (Popup)*

To enable the field list in pivot table UI, set the [ShowFieldList](#) property in [PivotView](#) class to **true**. A small icon will appear on the top left corner of the pivot table and clicking on this icon, field list dialog will appear.

**Note:** The field list icon will be displayed at the top right corner of the pivot table, when grouping bar is enabled.

#### CSHTML

```
@model List<PivotTableSample.Controllers.PivotData>
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Render()
```

#### FIELDLIST.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
public List<PivotData> GetPivotData()
{
 List<PivotData> pivotData = new List<PivotData>();
 pivotData.Add(new PivotData { Sold = 31, Amount = 52824, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
 pivotData.Add(new PivotData { Sold = 51, Amount = 86904, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
 pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country = "France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
}
```



```

pivotData.Add(new PivotData { Sold = 25, Amount = 42600, Country =
"France", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 27, Amount = 46008, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 49, Amount = 83496, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 95, Amount = 161880, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"France", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 75, Amount = 127800, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 67, Amount = 114168, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 69, Amount = 117576, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 90, Amount = 153360, Country =
"France", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 16, Amount = 27264, Country =
"France", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 83, Amount = 124422, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 57, Amount = 85448, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 20, Amount = 29985, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 67, Amount = 70008, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 89, Amount = 60496, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 75, Amount = 801880, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 57, Amount = 204168, Country =
"France", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 75, Amount = 737800, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 87, Amount = 884168, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 39, Amount = 729576, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 90, Amount = 38860, Country =
"France", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 93, Amount = 139412, Country =
"France", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 51, Amount = 92824, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 61, Amount = 76904, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 43360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 85, Amount = 62600, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 97, Amount = 86008, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 69, Amount = 93496, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q2" });

```

```

pivotData.Add(new PivotData { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 49, Amount = 89576, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 40, Amount = 33360, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 96, Amount = 77264, Country =
"Germany", Products = "Mountain Bikes", Year = "FY 2018", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 23, Amount = 24422, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 67, Amount = 75448, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 52345, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 13, Amount = 135612, Country =
"Germany", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 57, Amount = 90008, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 29, Amount = 90496, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 45, Amount = 301880, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 77, Amount = 404168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 15, Amount = 137800, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1" });
pivotData.Add(new PivotData { Sold = 37, Amount = 184168, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2" });
pivotData.Add(new PivotData { Sold = 99, Amount = 829576, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3" });
pivotData.Add(new PivotData { Sold = 80, Amount = 38360, Country =
"Germany", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q4" });
pivotData.Add(new PivotData { Sold = 91, Amount = 67824, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 81, Amount = 99904, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q2" });
pivotData.Add(new PivotData { Sold = 70, Amount = 49360, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q3" });
pivotData.Add(new PivotData { Sold = 65, Amount = 69600, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2015", Quarter =
"Q4" });
pivotData.Add(new PivotData { Sold = 57, Amount = 90008, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q1" });
pivotData.Add(new PivotData { Sold = 29, Amount = 90496, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q2" });

```

```

 pivotData.Add(new PivotData { Sold = 85, Amount = 391880, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 97, Amount = 904168, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2016", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 85, Amount = 237800, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 77, Amount = 384168, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q2" });
 pivotData.Add(new PivotData { Sold = 99, Amount = 829576, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q3" });
 pivotData.Add(new PivotData { Sold = 80, Amount = 38360, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2017", Quarter =
"Q4" });
 pivotData.Add(new PivotData { Sold = 76, Amount = 97264, Country =
"United States", Products = "Mountain Bikes", Year = "FY 2018", Quarter =
"Q1" });
 pivotData.Add(new PivotData { Sold = 53, Amount = 94422, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 90, Amount = 45448, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 29, Amount = 92345, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 67, Amount = 235612, Country =
"United States", Products = "Road Bikes", Year = "FY 2015", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 97, Amount = 90008, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 79, Amount = 90496, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 95, Amount = 501880, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q3"
});
 pivotData.Add(new PivotData { Sold = 97, Amount = 104168, Country =
"United States", Products = "Road Bikes", Year = "FY 2016", Quarter = "Q4"
});
 pivotData.Add(new PivotData { Sold = 95, Amount = 837800, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q1"
});
 pivotData.Add(new PivotData { Sold = 87, Amount = 684168, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q2"
});
 pivotData.Add(new PivotData { Sold = 109, Amount = 29576, Country =
"United States", Products = "Road Bikes", Year = "FY 2017", Quarter = "Q3"
});
 return pivotData;
 }
}
public class PivotData

```

```
{
 public int Sold { get; set; }
 public double Amount { get; set; }
 public string Country { get; set; }
 public string Products { get; set; }
 public string Year { get; set; }
 public string Quarter { get; set; }
}
```

<!-- markdownlint-disable MD012 -->

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

<br/>

Field List

All Fields

- ☒ Country
- ☒ Products
- ☒ Quarter
- ☒ Sold Amount
- ☒ Units Sold
- ☒ Year

Filters

Drop filter here

Rows

- Country
- Products

Columns

- Year
- Quarter

Values

- Sum of Units Sold
- Sum of Sold Amount

CLOSE

*Stand-alone Field List (Fixed)*

The field list can be rendered in a static position, anywhere in web page layout, like a separate component. To do so, you need to set [RenderMode](#) property to [Mode.Fixed](#) in [PivotFieldList](#).

**Note:** To make field list interact with pivot table, you need to use the **UpdateView** and **Update** methods for data source update in both field list and pivot table simultaneously.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEnginePopulate").Render()

@Html.EJS().PivotFieldList("Static_FieldList").RenderMode(Mode.Fixed).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).EnginePopulated("onFieldListEnginePopulate").Render()
<style>
 #Static_FieldList {
 width: 400px;
 }
</style>
<script>
 var pivotObj; var fieldlistObj;
 function onGridEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 if (fieldlistObj) {
 fieldlistObj.update(pivotObj);
 }
 }
 function onFieldListEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 fieldlistObj.updateView(pivotObj);
 }
</script>
```

**STATIC.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             |
|---------------|------------|-------------|------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |

**Pivot Field List**

All Fields

- ☒ Quarter
- ☒ Full Year
- ☒ Products
- ☒ Country
- ☒ Sold Amount

Drag fields between axes below:

Filters

Columns

- Full Year
- Quarter

Rows

- Country
- Products

Values

- Sum of Units Sold
- Sum of Sold Am...

*Invoking dynamic Field List (Customized)*

Also, you can display the field list dialog independently through other means. For example, you can invoke the field list dialog on an external button click. To do so, set [RenderMode](#) property to [Mode.Popup](#) and since on button click, field list dialog will be invoked.

**Note:** \* Meanwhile, you can display the field list dialog at specific target element within a webpage using [target](#) property. By default, the [target](#) value is null, which refers the [document.body](#) element.

\* Moreover, to make field list interact with pivot table, you need to use the **updateView** and **update** methods for data source update in both field list and pivot table simultaneously.

The below sample code illustrates the field list dialog invoked on an external button click.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("fieldlistbtn").Content("Field
List").IsPrimary(true).Render()
<div id="Popup_FieldList"></div>
@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEngi
nePopulate").Render()
```

```

@Html.EJS().PivotFieldList("PivotFieldList").RenderMode(Syncfusion.EJ2.Pivot
View.Mode.Popup).Target("#Popup_FieldList").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).EnginePopulated("onFieldListEnginePopulate").Render()
<style>
.e-toggle-field-list {
display: none !important;
}
</style>
<script>
var pivotObj; var fieldlistObj;
function onGridEnginePopulate(args) {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
if (fieldlistObj) {
fieldlistObj.update(pivotObj);
}
}
function onFieldListEnginePopulate(args) {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
fieldlistObj.updateView(pivotObj);
}
document.getElementById('fieldlistbtn').onclick = function () {
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
fieldlistObj.dialogRenderer.fieldListDialog.show();
};
</script>

```

### POPUP.CS

```

public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```

FIELD LIST

|               | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |             | Grand Total |
|---------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  |
| France        | 450        | \$714,955   | 525        | \$1,542,104 | 592        | \$2,983,308 | 76         | \$27,264    |             |
| Germany       | 440        | \$563,515   | 495        | \$1,772,104 | 372        | \$1,634,808 | 96         | \$77,264    |             |
| United States | 545        | \$754,515   | 636        | \$2,263,104 | 632        | \$3,041,448 | 76         | \$97,264    |             |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1596       | \$7,579,564 | 188        | \$201,792   |             |

### Search desired field

End user can search for desired field in the field list UI by typing the field name into the search box at runtime. It can be enabled by setting the [enableFieldSearching](#) property to **true** via code-behind.

**Note:** By default, field search option is disabled in the field list UI.

To enable search box in the static field list UI, set the [enableFieldSearching](#) property to **true** in [PivotFieldList](#).

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEnginePopulate").Render()

@Html.EJS().PivotFieldList("Static_FieldList").RenderMode(Mode.Fixed).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
).EnginePopulated("onFieldListEnginePopulate").EnableFieldSearching(true).Render()

<style>
 #Static_FieldList {
 width: 400px;
 }
</style>
<script>
 var pivotObj; var fieldlistObj;
```



```

function onGridEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 if (fieldlistObj) {
 fieldlistObj.update(pivotObj);
 }
}
function onFieldListEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 fieldlistObj.updateView(pivotObj);
}
</script>

```

### SEARCH.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

![output](images/Search desired field in static field list.png "Searching Static FieldList")

To enable search box in the pivot table's built-in popup field list UI, set the [enableFieldSearching](#) property to **true** in [PivotView](#).

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("fieldlistbtn").Content("Field
List").IsPrimary(true).Render()
<div id="Popup_FieldList"></div>
@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEngi
nePopulate").EnableFieldSearching(true).Render()
@Html.EJS().PivotFieldList("PivotFieldList").RenderMode(Syncfusion.EJ2.Pivot
View.Mode.Popup).Target("#Popup_FieldList").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true).EnableFieldSearch(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{

```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).EnginePopulated("onFieldListEnginePopulate").Render()
<style>
 .e-toggle-field-list {
 display: none !important;
 }
</style>
<script>
 var pivotObj; var fieldlistObj;
 function onGridEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 if (fieldlistObj) {
 fieldlistObj.update(pivotObj);
 }
 }
 function onFieldListEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 fieldlistObj.updateView(pivotObj);
 }
 document.getElementById('fieldlistbtn').onclick = function () {
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 fieldlistObj.dialogRenderer.fieldListDialog.show();
 };
</script>

```

### SEARCH.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

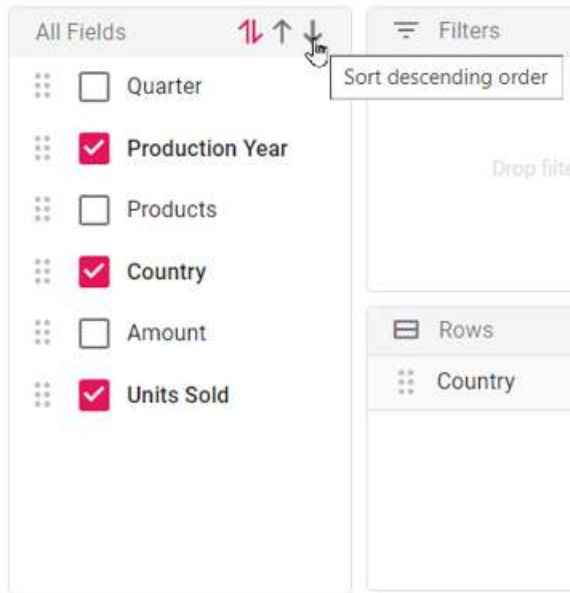
![output](images/Search desired field in popup field list.png "Searching Popup FieldList")

#### Option to sort fields

End user can sort fields in the field list UI to ascending (or) descending (or) default order (as obtained from the data source) using the built-in sort icons.

By default, fields are displayed in the default order.

## Field List



## Sort fields in a desired order

To display the fields in descending order by default, set the `DefaultFieldListOrder` property to **Descending** in the [Load](#) event.

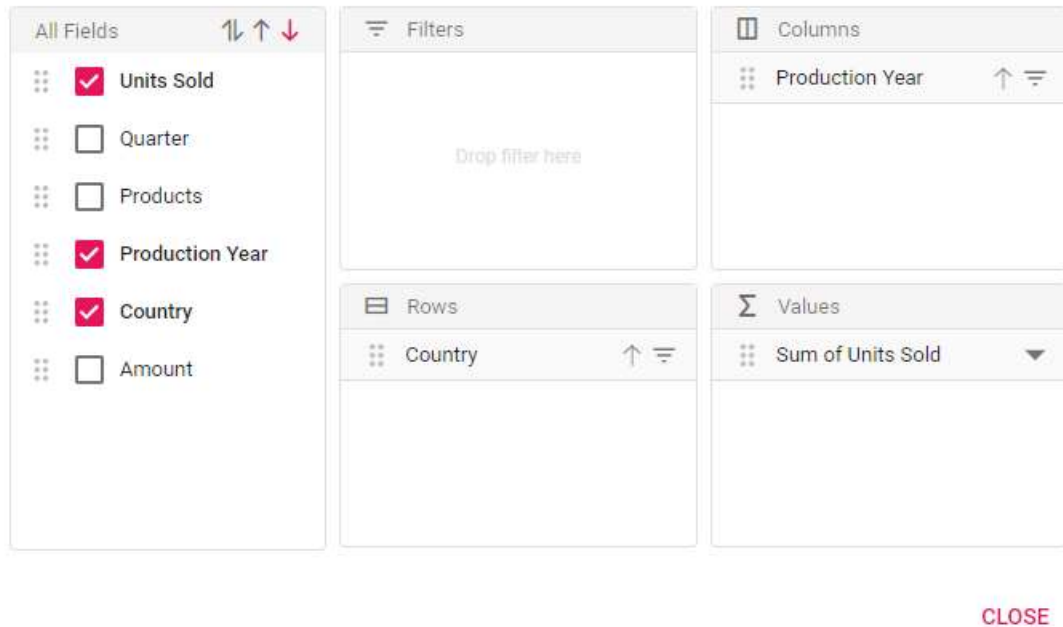
**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(350).Load("onLoad").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true)
.AllowValueFilter(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Production Year").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
})).ShowFieldList(true).ShowGroupingBar(true).Render()
<script>
 function onLoad(args) {
 args.defaultFieldListOrder = 'Descending';
 }
</script>
```

**FIELD SORT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Field List



### Group fields under desired folder name

In the field list UI, you can display fields by grouping them under the desired folder name. It can only be configured via code-behind by setting the [GroupName](#) property in [FieldMapping](#).

**Note:** You can only group fields to one level using the [GroupName](#) property.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(340).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add();
}).Columns(columns =>
{
 }
```

```

 columns.Name("Year").Caption("Production Year").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 }).FieldMapping(fields=>
 {
 fields.Name("Quarter").GroupName("Product category").Add();
 fields.Name("Products").GroupName("Product category").Add();
 fields.Name("Amount").GroupName("Product category").Caption("Sold
Amount").Add();
 }).ShowGroupingBar(true).ShowFieldList(true).Render()

```

### GROUPNAME.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Field List

The screenshot displays a 'Field List' interface with four main panels:

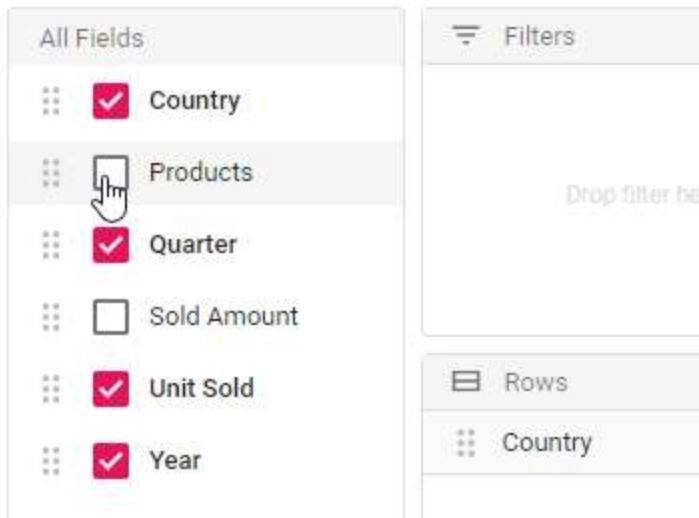
- All Fields:** A list of fields with checkboxes for selection. Under 'Product category', there are 'Quarter', 'Products', and 'Sold Amount' (all unchecked). Below this, 'Production Year', 'Country', and 'Units Sold' are checked with red checkmarks.
- Filters:** A panel with a 'Drop filter here' instruction and a filter icon.
- Columns:** A panel showing 'Production Year' as the column field, with an upward arrow and a filter icon.
- Rows:** A panel showing 'Country' as the row field, with an upward arrow and a filter icon.
- Values:** A panel showing 'Sum of Units Sold' as the value field, with a dropdown arrow.

CLOSE

### Add or remove fields

Using check box besides each field, end user can select or unselect to add or remove fields respectively from the report at runtime.

## Field List



### *Remove specific field(s) from displaying*

When a data source is bound to the component, fields will be automatically populated inside the Field List. In such case, user can also restrict specific field(s) from displaying. To do so, set the appropriate field name(s) in [ExcludeFields](#) property belonging to [PivotViewDataSourceSettings](#) class.

### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ExcludeFields(ViewBag.Excludefields).ShowFieldList(true).Render()
```

### **FIELDLIST.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
```

```
ViewBag.DataSource = data;
ViewBag.Excludefields = new string[] { "Products", "Amount" };
return View();
}
```

Field List

All Fields

☒

Country

☒

Quarter

☒

Unit Sold

☒

Year

Filters

Drop filter here

Rows

Country

Re-arranging fields

In-order to re-arrange, drag any field from the field list and drop it into the column, row, value, or filter axis using the drag-and-drop holder. It helps end user to alter the report at runtime.

Filters

Products (All)

Columns

Year

Quarter

Rows

Country

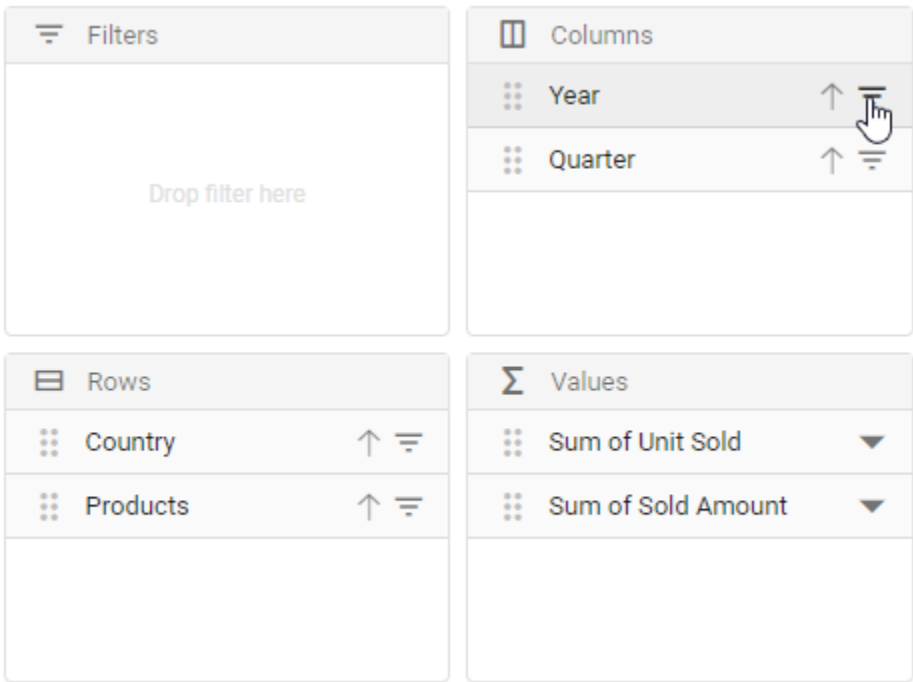
Values

Sum of Unit Sold

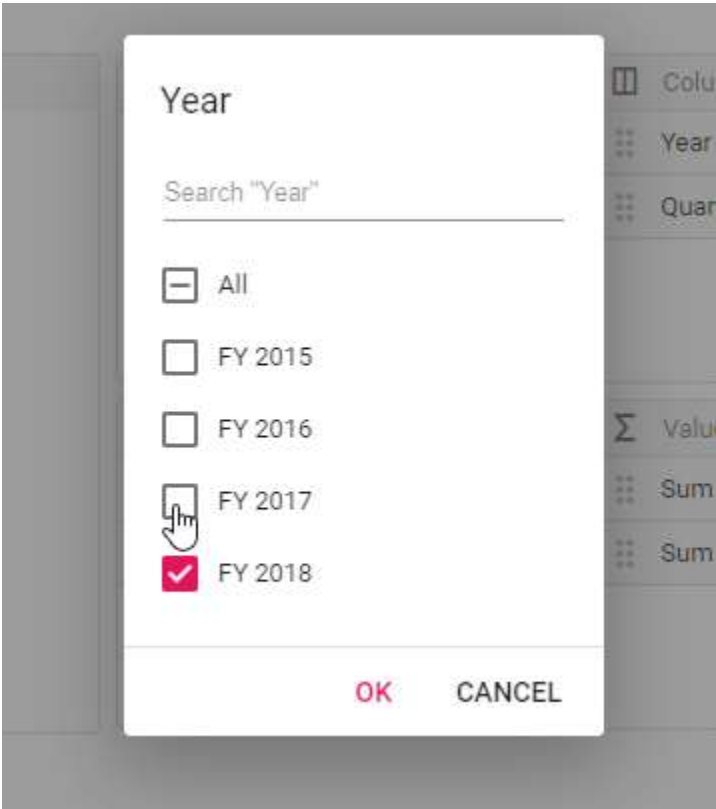
Sum of Sold Amount

Filtering members

Using the filter icon besides each field in row, column and filter axes, members can be either included or excluded at runtime. To know more about member filtering, [refer](#) here.



<br/>





<br/>

|               | FY 2018    |             | Grand Total |             |
|---------------|------------|-------------|-------------|-------------|
|               | Units Sold | Sold Amount | Units Sold  | Sold Amount |
| France        | 16         | \$27,264    | 16          | \$27,264    |
| Germany       | 96         | \$77,264    | 96          | \$77,264    |
| United States | 76         | \$97,264    | 76          | \$97,264    |
| Grand Total   | 188        | \$201,792   | 188         | \$201,792   |

Sorting members

Using the sort icon besides each field in row and column axes, members can be arranged either in ascending or descending order at runtime. To know more about member sorting, [refer](#) here.

Filters

Drop filter here

Columns

Year

Quarter

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

<br/>

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Calculated fields

The calculated field support allows end user to add a new calculated field based on the available fields from the bound data source using basic arithmetic operators. To enable this support in Field List UI, set the [AllowCalculatedField](#) property in [PivotView](#) class to **true** in pivot table. Now a button will be seen automatically inside the field list UI which will invoke the calculated field dialog on click. To know more about calculated field, [refer](#) here.

### Field List

CALCULATED FIELD

All Fields

Country

✓

Products

✓

Quarter

✓

Sold Amount

✓

Total Amount

✓

Unit Sold

✓

Year

✓

Filters

Drop filter here

Rows

Country

↑

☰

Products

↑

☰

Columns

Year

↑

☰

Quarter

↑

☰

Values

Sum of Unit Sold

▼

Sum of Sold Amount

▼

Total Amount

CLOSE

&lt;br/&gt;

Create Calculated Field

×

Total Amount

---

Drag and drop fields to formula

⋮

Total Amount (CalculatedField)

✎

⋮

Unit Sold (Sum)

>

⋮

Year (Count)

Formula

"Sum(Amount)\*"Sum(Sold)"

OK

CANCEL

&lt;br/&gt;

|               | FY 2015    |             |              | FY 2016    |             |
|---------------|------------|-------------|--------------|------------|-------------|
|               | Units Sold | Sold Amount | Total Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | \$715,405    | 526        | \$1,542,    |
| Germany       | 440        | \$563,515   | \$563,955    | 496        | \$1,772,    |
| United States | 546        | \$754,515   | \$755,061    | 636        | \$2,263,    |
| Grand Total   | 1436       | \$2,032,985 | \$2,034,421  | 1658       | \$5,577,    |

#### Changing aggregation type of value fields at runtime

End user can perform calculations over a group of values using the aggregation option. The value fields bound to the field list, appears with a dropdown icon, helps to select an appropriate aggregation type at runtime. On selection, the values in the Pivot Table will be changed dynamically. To know more about aggregation, [refer](#) here.

Filters

Drop filter here

Columns

Year

Quarter

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

<br/>

<br/>

Rows

Country

Products

Values

Sum of Unit Sold

Sum of Sold Amount

Sum

Count

Distinct Count

Product

Avg

Min

Max

More...

<br/>

|               | FY 2015        |             | FY 2016        |             | FY 2017    |
|---------------|----------------|-------------|----------------|-------------|------------|
|               | Units Sold     | Sold Amount | Units Sold     | Sold Amount | Units Sold |
| United States | 68.25          | \$754,515   | 79.5           | \$2,263,104 | 91.        |
| Germany       | 55             | \$563,515   | 62             | \$1,772,104 |            |
| France        | 56.25          | \$714,955   | 65.75          | \$1,542,104 |            |
| Grand Total   | 59.83333333... | \$2,032,985 | 69.08333333... | \$5,577,312 | 70.        |

### Defer layout update

Defer layout update support to update the pivot table only on demand and not during every user action. To enable this support in Field List UI, set the [AllowDeferLayoutUpdate](#) property in [PivotView](#) class to **true** in pivot table. Now a check box inside Field List UI will be seen in checked state, allowing pivot table to update only on demand. To know more about defer layout, [refer](#) here.

☒ Unit Sold  
☒ Year

Rows

Country ↑

Products ↑

Σ Values

Sum of Unit Sold ▼

Sum of Sold Amount ▼

☒ Defer Layout Update

APPLY

CANCEL

### Show built-in Field List (Popup) over specific target

By passing the target element to the built-in field list dialog module in the [dataBound](#) event, the field list dialog will be displayed over the appropriate target element on a web page. By default, the Pivot Table's parent element is used as the target element to display the built-in field list dialog.

The sample code below shows the built-in field list dialog using `document.body` as the target element.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
dataSourceSettings =>
{
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false);
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
```

```

{
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})).DataBound("ondataBound").ShowFieldList(true).Render()
<script>
 function ondataBound(args) {
 var pivotTableObj =
 document.getElementById('PivotView').ej2_instances[0];

 pivotTableObj.pivotFieldListModule.dialogRenderer.fieldListDialog.target =
 document.body;
 }
</script>

```

### FIELDLIST-ON-SPECIFICTARGET.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             |
|---------------|------------|-------------|------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |

#### Show field list using toolbar

It can also be viewed in toolbar by setting [ShowFieldList](#) and [ShowToolbar](#) properties in [PivotView](#) class to **true**. Also, include the item **FieldList** within the [Toolbar](#) property in [PivotView](#) class. When toolbar is enabled, field list icon will be automatically added into the toolbar and the icon won't appear on top left corner in the pivot table component.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>

```

```

dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).Toolbar(new List<string>() { "FieldList" }).Render()

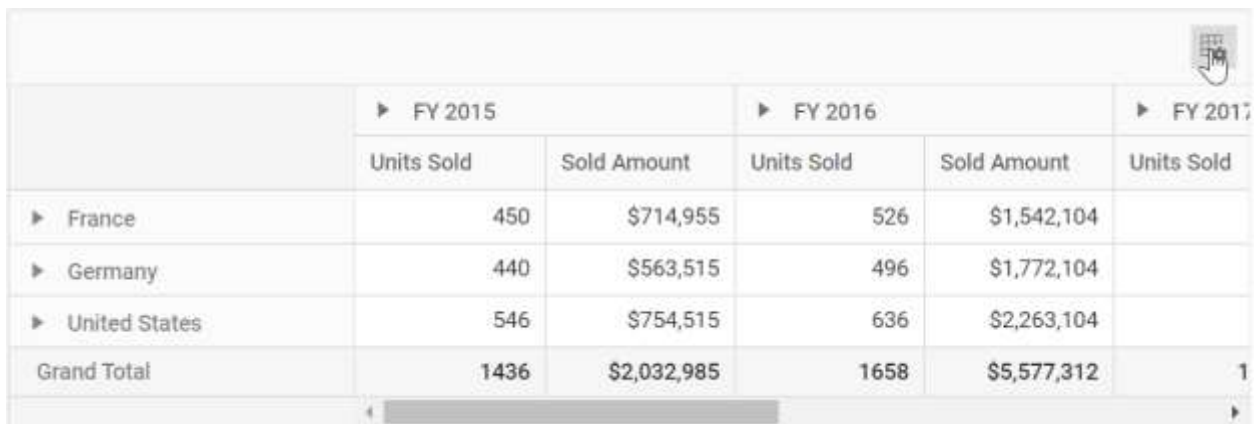
```

### FIELDLIST.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

*Set caption to fields which isn't bound to the report*

One can set the caption to all fields from the data source even if it is not bound to the actual report. It can be achieved using the [EnginePopulated](#) event. On doing so, caption of the respective field will be displayed in both grouping bar and field list.

In the sample, we have set caption to the fields **Year** and **Quarter** dynamically.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEnginePopulate").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .FormatSettings(formatsettings =>
 {

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }) .Rows(rows =>
 {
 rows.Name("Country").Add();
 }) .Columns(columns =>
 {
 columns.Name("Products").Add();
 }) .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }) .ShowFieldList(true).Render()
<script>
var pivotObj;
function onGridEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 Object.keys(pivotObj.engineModule.fieldList).forEach((key, index)
=> {
 if (key === 'Quarter') {
 pivotObj.engineModule.fieldList[key].caption =
'Production Quarter Year';
 }
 else if (key === 'Year') {
 pivotObj.engineModule.fieldList[key].caption =
'Production Year';
 }
 });
}
</script>

```

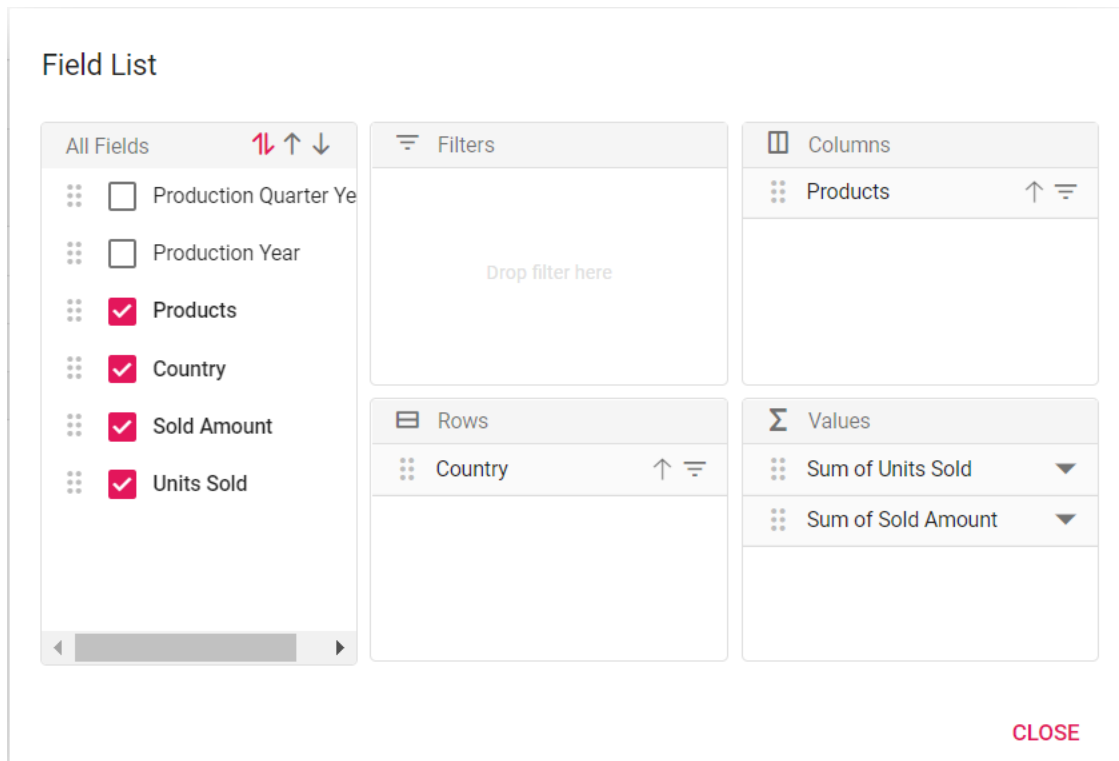
### FIELDLIST.CS

```

public IActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```





During runtime, the **Values** button in the field list can be moved to a different position (i.e., different index) among other fields in the column or row axis. To enable the **Values** button, set the [showValuesButton](#) property to **true**.

**Note:** This support is only available for relational data sources.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEnginePopulate").Render()

@Html.EJS().PivotFieldList("PivotFieldList").RenderMode(Mode.Fixed).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();

 })).EnginePopulated("onFieldListEnginePopulate").ShowValuesButton(true).Render()
</style>
 #Static_FieldList {
 width: 400px;
 }
</style>
<script>
 var pivotObj; var fieldlistObj;
 function onGridEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 if (fieldlistObj) {
 fieldlistObj.update(pivotObj);
 }
 }
 function onFieldListEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 fieldlistObj.updateView(pivotObj);
 }
</script>

```

#### **FIELDLIST-VALUESBUTTON.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Pivot Field List

All Fields
↕ ↑ ↓

☰

☒ Quarter

☰

☒ Year

☰

☒ Products

☰

☒ Country

☰

☒ Sold Amount

Drag fields between axes below:

☰ Filters

Drop filter here

☰ Columns

☰ Year

↑ ☰

☰ Quarter

↑ ☰

☰ Values

☰ Rows

☰ Country

↑ ☰

☰ Products

↑ ☰

Σ Values

☰ Sum of Units Sold

▼

☰ Sum of Sold Amount

▼

#### Events

##### EnginePopulated

The [EnginePopulated](#) event is available in both Pivot Table and Field List.

- The event [EnginePopulated](#) is triggered in field list whenever the report gets modified. The updated report is passed to the pivot table via `UpdateView` method written within this event to refresh the same.
- Likewise, [EnginePopulated](#) event is triggered in pivot table whenever the report gets modified. The updated report is passed to the field list via `Update` method written within this event to refresh the same.

The event [EnginePopulated](#) is triggered after engine is populated. It has following parameters - `DataSourceSettings`, `PivotFieldList` and `PivotValues`.

**Note:** This event is not required for Popup field list since it is a in built one.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").EnginePopulated("onGridEnginePopulate").Render()

@Html.EJS().PivotFieldList("Static_FieldList").RenderMode(Mode.Fixed).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).EnginePopulated("onFieldListEnginePopulate").Render()
<style>
#Static_FieldList {
width: 400px;
}
</style>
<script>
var pivotObj; var fieldlistObj;
function onGridEnginePopulate(args) {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
if (fieldlistObj) {
fieldlistObj.update(pivotObj);
}
}
function onFieldListEnginePopulate(args) {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
fieldlistObj.updateView(pivotObj);
}
</script>
```

### STATIC.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### FieldListRefreshed

The event [FieldListRefreshed](#) is triggered whenever there is any change done in the field list UI. It has following parameter - [DataSourceSettings](#) and [PivotValues](#). It allows user to identify each field list update. This event is applicable only for static field list.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotGrid").Height("300").DataSource(dataSource =>
dataSource.Data((IEnumerable<object>)ViewBag.Data).FieldListRefreshed("field
ListRefreshed").ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).
AllowValueFilter(true)
 .FormatSettings(formatsettings =>
 {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowFieldList(true).Render()
<script>
 function fieldListRefreshed(args) {
 //Triggers, whenever field list get refreshed.
 }
</script>
```

### POPUP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### FieldDropped

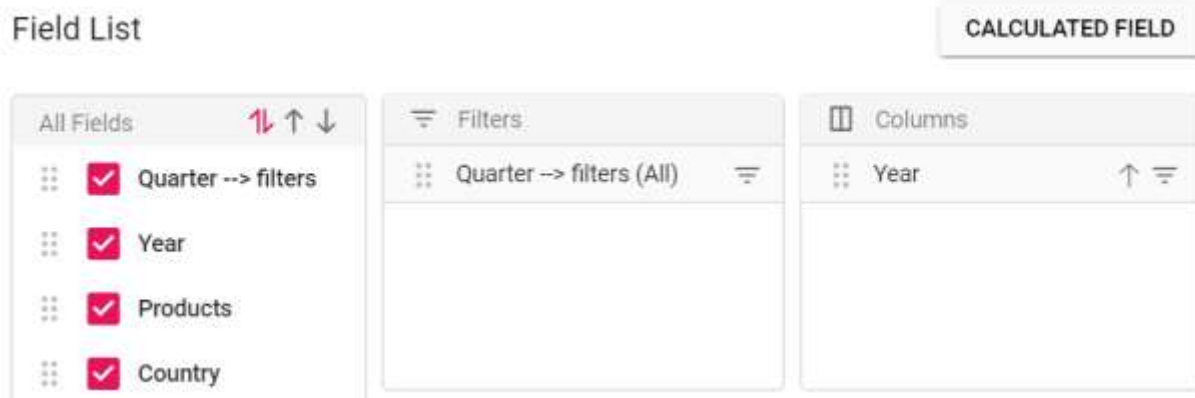
The event [OnFieldDropped](#) fires whenever a field is dropped in an axis. It has following parameters - **DroppedAxis**, **DroppedField** and **DataSourceSettings**. In this illustration, we have modified the **DroppedField** caption through this event at runtime.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource
ce =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
).ShowFieldList(true).OnFieldDropped("onFieldDropped").Render()
<script>
var pivotGridObj;
function onFieldDropped(args) {
args.droppedField.caption = args.droppedField.name + " --> " +
args.droppedAxis;
}
</script>
```

### DROPPED.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.data = data;
return View();
}
```



### ActionBegin

The event [actionBegin](#) triggers when the UI actions such as sorting, filtering, aggregation or edit calculated field, that are present in the field list UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

|                                             |                                                   |
|---------------------------------------------|---------------------------------------------------|
| Action                                      | Action Name                                       |
| -----                                       | -----                                             |
| <a href="#">Sort icon</a>                   | Sort field                                        |
| <a href="#">Filter icon</a>                 | Filter field                                      |
| <a href="#">Aggregation</a>                 | (Value type drop down and menu)   Aggregate field |
| <a href="#">Edit icon</a>                   | Edit calculated field                             |
| <a href="#">Calculated field button</a>     | Open calculated field dialog                      |
| <a href="#">Field list</a>                  | Open field list                                   |
| <a href="#">Field list tree – Sort icon</a> | Sort field tree                                   |

- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **cancel**: It allows user to restrict the current action.

In the below sample, opening pop-up field list can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
).ShowFieldList(true).ActionBegin("actionBegin").Render()
<script>
function actionBegin(args) {
 if (args.actionName == 'Open field list') {
 args.cancel = true;
 }
}
</script>

```

### ACTIONBEGIN-FIELDLIST.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### ActionComplete

The event [actionComplete](#) triggers when the UI actions such as sorting, filtering, aggregation or edit calculated field, that are present in the field list UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action | Action Name |
|--------|-------------|
| -----  | -----       |



- | [Sort icon](#) | Field sorted |
- | [Filter icon](#) | Field filtered |
- | [Aggregation](#) (Value type drop down and menu) | Field aggregated |
- | [Edit icon](#) | Calculated field edited |
- | [Calculated field button](#) | Calculated field applied |
- | [Field list](#) | Field list closed |
- | [Field list tree – Sort icon](#) | Field tree sorted |

- **fieldInfo**: It holds the selected field information.

**Note:** This option is applicable only when the field based UI actions are performed such as filtering, sorting, removing field from grouping bar, editing and aggregation type change.

- **actionInfo**: It holds the unique information about the current UI action. For example, if sorting is completed, the event argument contains information such as sort order and the field name.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
).ShowFieldList(true).ActionComplete("actionComplete").Render()
<script>
function actionComplete(args) {
 if (args.actionName == 'Field list closed') {
 // Triggers when the field list dialog is closed.
 }
}
</script>
```

**ACTIONCOMPLETE-FIELDLIST.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**ActionFailure**

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- **actionName**: It holds the name of the current action failed. The following are the UI actions and their names:

| Action | Action Name |

|-----|-----|

| [Sort icon](#) | Sort field |

| [Filter icon](#) | Filter field |

| [Aggregation](#) (Value type drop down and menu) | Aggregate field |

| [Edit icon](#) | Edit calculated field |

| [Calculated field button](#) | Open calculated field dialog |

| [Field list](#) | Open field list |

| [Field list tree – Sort icon](#) | Sort field tree |

- **errorInfo**: It holds the error information of the current UI action.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
```

```

{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).ShowFieldList(true).ActionFailure("actionFailure").Render()
<script>
 function actionFailure(args) {
 if (args.actionName == 'Open field list') {
 // Triggers when the current UI action fails to achieve the
 desired result.
 }
 }
}
</script>

```

### ACTIONFAILURE-FIELDLIST.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### See Also

- [Change load limited data in member editor](#)
- [Customize the icons for pivot table](#)

#### Defer Layout Update

Defer layout update support allows to update the pivot table component only on demand. On enabling this feature, end user can drag-and-drop fields between row, column, value and filter axes, apply sorting and filtering inside the Field List, resulting in change of pivot report alone but not the pivot table values. Once all operations are performed and on clicking the "Apply" button in the Field List, pivot table will start to update with the last modified report. This also helps in bringing better performance in pivot table component rendering.

The field list can be displayed in two different formats to interact with pivot table. They are:

- **In-built Field List (Popup):** To display the field list icon in pivot table UI to invoke the built-in dialog.
- **Stand-alone Field List (Fixed):** To display the field list in a static position within a web page.

#### *In-built Field List (Popup)*

To enable deferred updates in the pivot table, set the property [AllowDeferLayoutUpdate](#) in [PivotView](#) as **true**. To make a note, the defer update option can be controlled only via Field List during runtime.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>

```

```

dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).AllowDeferLayoutUpdate(true).Render()

```

### DEFERUPDATE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                                                                                           |                                                            |                                                                                   |
|-------------------------------------------------------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------|
| <input checked="" type="checkbox"/> Unit Sold<br><input checked="" type="checkbox"/> Year | <div>Rows</div> <div>Country ↑</div> <div>Products ↑</div> | <div>Σ Values</div> <div>Sum of Unit Sold ▼</div> <div>Sum of Sold Amount ▼</div> |
|-------------------------------------------------------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------|

☒ Defer Layout Update

APPLY

CANCEL

### Stand-alone Field List (Fixed)

The field list can be rendered in a static position, anywhere in web page layout, like a separate component. To do so, you need to set [RenderMode](#) property to [Mode.Fixed](#) in [PivotFieldList](#).

To enable deferred updates in the static fieldlist, set the property [AllowDeferLayoutUpdate](#) in [PivotFieldList](#) as **true**. To make a note, the defer update option can be controlled only via Field List during runtime.

**Note:** To make field list interact with pivot table, you need to use the **UpdateView** and **Update** methods for data source update in both field list and pivot table simultaneously.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").AllowDeferLayoutUpdate(true)
.EnginePopulated("onGridEnginePopulate").Render()

@Html.EJS().PivotFieldList("Static_FieldList").RenderMode(Mode.Fixed).DataSourceSettings(
dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1)
.UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.EnginePopulated("onFieldListEnginePopulate").AllowDeferLayoutUpdate(true)
.Render()
<style>
#Static_FieldList {
width: 400px;
}
</style>
<script>
var pivotObj; var fieldlistObj;
function onGridEnginePopulate(args) {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
if (fieldlistObj) {
fieldlistObj.update(pivotObj);
}
}
function onFieldListEnginePopulate(args) {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
fieldlistObj =
document.getElementById('PivotFieldList').ej2_instances[0];
if (fieldlistObj.isRequiredUpdate) {
fieldlistObj.updateView(pivotObj);
}
pivotObj.notify('ui-update', pivotObj);
fieldlistObj.notify('tree-view-update', fieldlistObj);
}
</script>

```

### STATIC.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             |
|---------------|------------|-------------|------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | 526        |             |
| Germany       | 440        | \$563,515   | 496        |             |
| United States | 546        | \$754,515   | 636        |             |
| Grand Total   | 1436       | \$2,032,985 | 1658       |             |

### Pivot Field List

All Fields

- ☒ Quarter
- ☒ Year
- ☒ Products
- ☒ Country
- ☒ Sold Amount

Drag fields between axes below:

Filters

Drag-able Area

Columns

- Year
- Quarter

Rows

- Country
- Products

Values

- Sum of Units Sold
- Sum of Sold Amo...

☒ Defer Layout Update **APPLY** **CANCEL**

## Performance

<!-- markdownlint-disable MD036 -->

### Virtual Scrolling

#### Virtual Scrolling

The virtual scrolling option allows you to load the large amounts of data without performance degradation by rendering rows and columns only in the content viewport. The data will refresh dynamically on vertical or horizontal scroll. This feature can be enabled by setting the [EnableVirtualization](#) property in [PivotView](#) class to **true**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(300).Width(800).DataSourceSettings
(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(true)
).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
```

```

columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
}))).EnableVirtualization(true).Render()

```

### VIRTUALSCROLLING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                | ▼ FY 2015  |             |            |             |           |
|----------------|------------|-------------|------------|-------------|-----------|
|                | Q1         |             | Q2         |             | Q3        |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sol |
| ▼ France       | 114        | \$177,246   | 108        | \$172,352   |           |
| Mountain Bikes | 31         | \$52,824    | 51         | \$86,904    |           |
| Road Bikes     | 83         | \$124,422   | 57         | \$85,448    |           |
| ▼ Germany      | 74         | \$117,246   | 128        | \$152,352   |           |
| Mountain Bikes | 51         | \$92,824    | 61         | \$76,904    |           |

### Limitations for virtual scrolling

- In virtual scrolling, the [ColumnWidth](#) property in [GridSettings](#) should be in pixel and percentage values are not accepted.
- Resizing columns, setting width to individual columns which affects the calculation used to pick the correct page on scrolling.
- Grouping, which takes additional time to splitting the raw items into the provided format.
- Date Formatting, which takes additional time to convert date format.
- Date Formatting with sorting, here additionally full date time format should be framed to perform sorting along with the provided date format which lags the performance.
- When using OLAP data, subtotals and grandtotals are only displayed when measures are bound at the last position in the [Row](#) or [Column](#) axis. Otherwise, the data from the pivot table will be shown without summary totals.
- Even if virtual scrolling is enabled, not only is the current view port data retrieved, but also the data for the immediate previous page and the immediate next page. As a result, when the end user scrolls slightly ahead or behind, the next or previous page data is displayed immediately without requiring a refresh. **Note:** If the pivot table's width and height are large, the loading data count in the current, previous, and next view ports (pages) will also increase, affecting performance.

### Data Compression

**Note:** This property is applicable only for relational data source.

When we bind one million raw data, the pivot table will process all raw data to generate aggregated data during initial rendering and report manipulation. But in data compression, the data will be compressed based on the uniqueness of the raw data, and unique records will be provided as input for the Pivot Table. The compressed data will be used for further operations at all times, reducing the looping complexity and improving the performance of the pivot table. For example, if the pivot table is connected to one million raw data aggregated to 1,000 unique data means, it will be rendered within 3 seconds rather than 10 seconds. You can enable this option by using the [AllowDataCompression](#) property along with [EnableVirtualization](#) property.

**Note:** This options will only function when the virtual scrolling is enabled.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(300).Width(800).DataSourceSettings
(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.EnableVirtualization(true).AllowDataCompression(true).Render()

```

### VIRTUALSCROLLING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Limitations during data compression

- The following aggregation types will not be supported.
- Average
- Populationstdev
- Samplestdev
- Populationvar
- Samplevar
- If you use any of the aggregations above, it will result in an aggregation type “Sum”.
- Distinctcount will act as “Count” aggregation type.



- In the calculated field, an existing field can be inserted without altering its default aggregation type. Even if we change it, it would use the default aggregation type back for calculation.

#### *Virtual scrolling for static field list*

Virtual scrolling automatically works with "Popup" field list on setting the [EnableVirtualization](#) property in the Pivot Table to **true**. In case of static field list, which act as a separate component, user need to enable [EnableVirtualization](#) property in the Pivot Table and also pass the report information to pivot table instance via the [load](#) event of the field list.

#### **CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
EnginePopulated("onGridEnginePopulate").Render()

@Html.EJS().PivotFieldList("PivotFieldList").RenderMode(Mode.Fixed).DataSourceSettings(dataSource =>
dataSource.DataSource(data(1000)).ExpandAll(false).EnableSorting(true)
.Rows(rows =>
{
 rows.Name("ProductID").Add();
}).Columns(columns =>
{
 columns => { columns.Name("Year").Add();
}).Values(values =>
{
 values.Name("Price").Caption("Unit Price").Add();
 values.Name("Sold").Caption("Unit Sold").Add();
}).Load("onLoad").EnginePopulated("onFieldListEnginePopulate").Render()
<style>
 #PivotView {
 width: 58%;
 height: 100%;
 }
 #PivotFieldList {
 width: 42%;
 height: 100%;
 }
 .e-PivotView {
 float: left;
 }
 .e-PivotFieldList {
 float: right;
 }
 .e-PivotFieldList .e-static {
 width: 100% !important;
 }
</style>
<script>
 var dt = 0;
 var data = function (count) {
 var result = [];
 for (var i = 1; i < (count + 1); i++) {
 dt++;
 var round = void 0;
 var toString_1 = i.toString();
```

```

 if (toString_1.length === 1) {
 round = '0000' + (i);
 }
 else if (toString_1.length === 2) {
 round = '000' + i;
 }
 else if (toString_1.length === 3) {
 round = '00' + i;
 }
 else if (toString_1.length === 4) {
 round = '0' + i;
 }
 else {
 round = toString_1;
 }
 result.push({
 ProductID: 'PRO-' + round,
 Year: "FY " + (dt + 2013),
 Price: Math.round(Math.random() * 5000) + 5000,
 Sold: Math.round(Math.random() * 80) + 10,
 });
 if (dt / 4 == 1)
 dt = 0;
 }
 return result;
};
var pivotObj; var fieldListObj;
function onGridEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldListObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 if (fieldListObj) {
 fieldListObj.update(pivotObj);
 }
}
function onFieldListEnginePopulate(args) {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldListObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 fieldListObj.updateView(pivotObj);
}
function onLoad(args) {
 //Getting component instance.
 fieldListObj =
document.getElementById('PivotFieldList').ej2_instances[0];
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 fieldListObj.pivotGridModule = pivotObj;
 //Assigning report to pivot table component.
 pivotObj.datasourcesettings = fieldListObj.datasourcesettings;
 //Generating page settings based on pivot table component's size.
 pivotObj.updatePageSettings(true);
 //Assigning page settings to field list component.
 fieldListObj.pageSettings = pivotObj.pageSettings;
}
</script>

```

**VIRTUALIZATION.CS**

```
public ActionResult Index()
{
 return View();
}
```

<!-- markdownlint-disable MD036 -->

**Paging in ASP.NET MVC Pivot Table Component**

Paging allows you to load large amounts of data that can be divided and displayed page by page in the pivot table. It can be enabled by setting the [EnablePaging](#) property to **true**. It can be configured at code-behind by using the [PageSettings](#) property, during initial rendering of the component. The properties required are:

- [CurrentRowPage](#): Allows user to set the current row page number to be displayed in the pivot table.
- [CurrentColumnPage](#): Allows user to set the current column page number to be displayed in the pivot table.
- [RowPageSize](#): Allows user to set the total number of records to be displayed on each page of the pivot table's row axis.
- [ColumnPageSize](#): Allows user to set the total number of records to be displayed on each page of the pivot table's column axis.

***Pager UI***

When paging is enabled, a built-in pager UI appears at the bottom of the pivot table, allowing you to change the current page in the row and column axes by navigating to a desired page using the navigation buttons or an input text box, as well as change the page size via dropdown at runtime.

You can also change the position, visibility, compact view, and template of the row and column pagers by using the [PagerSettings](#).

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("ProductName").Caption("Product Name").Add();
}).Values(values =>
{
```

```

 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
 })
).PageSettings(pageSettings =>
 pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings =>
 pagerSettings.Position(PagerPosition.Bottom).EnableCompactView(false).IsInversed(false).ShowColumnPager(true)

 .ShowRowPager(true).ShowRowPageSize(true).ShowColumnPageSize(true).RowPageSizes(ViewBag.RowPageSizes).ColumnPageSizes(ViewBag.ColumnPageSizes)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()

```

## PAGING.CS

```

public ActionResult Index()
{
 ViewBag.ColumnPageSizes = new double[] { 5, 10, 20, 50, 100 };
 ViewBag.RowPageSizes = new double[] { 10, 50, 100, 200 };
 return View();
}

```

|              | Boston Crab Meat |            | Camembert Pierrot |            | Guaraná Fantástica |            |
|--------------|------------------|------------|-------------------|------------|--------------------|------------|
|              | Quantity         | Unit Price | Quantity          | Unit Price | Quantity           | Unit Price |
| ▼ Argentina  | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| Buenos Aires | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| ▼ Austria    | 91               | \$18       | 160               | \$68       | 283                | \$22       |
| Graz         | 91               | \$18       | 160               | \$68       | 248                | \$17       |
| Salzburg     |                  |            |                   |            | 35                 | \$5        |
| ▼ Belgium    | 15               | \$37       | 40                | \$27       | 12                 | \$5        |
| Bruxelles    | 10               | \$18       |                   |            |                    |            |
| Charleroi    | 5                | \$18       | 40                | \$27       | 12                 | \$5        |
| ▼ Brazil     | 36               | \$66       | 212               | \$224      | 145                | \$26       |
| Campinas     |                  |            | 10                | \$34       |                    |            |

Row pager

1

of 9

10

Column pager

1

of 4

5

Columns per page

### Show pager UI at top or bottom

You can display the pager UI at top or bottom of the pivot table by using the [Position](#) property. To show the pager UI at top of the pivot table, set the [Position](#) property in [PagerSettings](#) to **Top**.

**Note:** By default, the pager UI appears at the bottom of the pivot table.

## CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{

```

```

dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {

formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
 }).Columns(columns =>
 {
 columns.Name("ProductName").Caption("Product Name").Add();
 }).Values(values =>
 {
 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
 })
).PageSettings(pageSettings =>
pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings =>
pagerSettings.Position(PagerPosition.Top)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()

```

## PAGING.CS

```

public ActionResult Index()
{
 return View();
}

```

| Row pager       |                  | Rows per page |                   | Column pager    |                    | Columns per page |  |
|-----------------|------------------|---------------|-------------------|-----------------|--------------------|------------------|--|
| < < > >  1 of 9 |                  | 10            |                   | < < > >  1 of 4 |                    | 5                |  |
|                 | Boston Crab Meat |               | Camembert Pierrot |                 | Guaraná Fantástica |                  |  |
|                 | Quantity         | Unit Price    | Quantity          | Unit Price      | Quantity           | Unit Price       |  |
| ▼ Argentina     | 20               | \$15          | 11                | \$19            | 36                 | \$35             |  |
| Buenos Aires    | 20               | \$15          | 11                | \$19            | 36                 | \$35             |  |
| ▼ Austria       | 91               | \$18          | 160               | \$68            | 283                | \$22             |  |
| Graz            | 91               | \$18          | 160               | \$68            | 248                | \$17             |  |
| Salzburg        |                  |               |                   |                 | 35                 | \$5              |  |
| ▼ Belgium       | 15               | \$37          | 40                | \$27            | 12                 | \$5              |  |
| Bruxelles       | 10               | \$18          |                   |                 |                    |                  |  |
| Charleroi       | 5                | \$18          | 40                | \$27            | 12                 | \$5              |  |
| ▼ Brazil        | 36               | \$66          | 212               | \$224           | 145                | \$26             |  |
| Campinas        |                  |               | 10                | \$34            |                    |                  |  |

[Inverse pager](#)

Toggles and displays row and column pager. To show the column pager on the left side of the pager UI, set the [IsInversed](#) property in [PagerSettings](#) to **true**.

**Note:** By default, the row pager is displayed on the left side of the pager UI, while the column pager is displayed on the right side.

**C#HTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("ProductName").Caption("Product Name").Add();
}).Values(values =>
{
 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
})
).PageSettings(pageSettings =>
pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings => pagerSettings.IsInversed(true)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()
```

**PAGING.CS**

```
public ActionResult Index()
{
 return View();
}
```

|              | Boston Crab Meat |            | Camembert Pierrot |            | Guaraná Fantástica |            |
|--------------|------------------|------------|-------------------|------------|--------------------|------------|
|              | Quantity         | Unit Price | Quantity          | Unit Price | Quantity           | Unit Price |
| ▼ Argentina  | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| Buenos Aires | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| ▼ Austria    | 91               | \$18       | 160               | \$68       | 283                | \$22       |
| Graz         | 91               | \$18       | 160               | \$68       | 248                | \$17       |
| Salzburg     |                  |            |                   |            | 35                 | \$5        |
| ▼ Belgium    | 15               | \$37       | 40                | \$27       | 12                 | \$5        |
| Bruxelles    | 10               | \$18       |                   |            |                    |            |
| Charleroi    | 5                | \$18       | 40                | \$27       | 12                 | \$5        |
| ▼ Brazil     | 36               | \$66       | 212               | \$224      | 145                | \$26       |
| Campinas     |                  |            | 10                | \$34       |                    |            |

Column pager

Columns per page

Row pager

Rows per page

|< < > >| 1 of 4
5

|< < > >| 1 of 9
10

### Compact view

By hiding all except the previous and next navigation buttons, the pager UI can be displayed with the absolute minimum of paging options. The compact view can be enabled by setting the [EnableCompactView](#) property in [PagerSettings](#) to **true**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("ProductName").Caption("Product Name").Add();
}).Values(values =>
{
 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
})
).PageSettings(pageSettings =>
pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings => pagerSettings.EnableCompactView(true)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()
```

**PAGING.CS**

```
public ActionResult Index()
{
 return View();
}
```

|                                                                        | Boston Crab Meat |            | Camembert Pierrot |            | Guaraná Fantástica |            |
|------------------------------------------------------------------------|------------------|------------|-------------------|------------|--------------------|------------|
|                                                                        | Quantity         | Unit Price | Quantity          | Unit Price | Quantity           | Unit Price |
| ▼ Argentina                                                            | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| Buenos Aires                                                           | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| ▼ Austria                                                              | 91               | \$18       | 160               | \$68       | 283                | \$22       |
| Graz                                                                   | 91               | \$18       | 160               | \$68       | 248                | \$17       |
| Salzburg                                                               |                  |            |                   |            | 35                 | \$5        |
| ▼ Belgium                                                              | 15               | \$37       | 40                | \$27       | 12                 | \$5        |
| Bruxelles                                                              | 10               | \$18       |                   |            |                    |            |
| Charleroi                                                              | 5                | \$18       | 40                | \$27       | 12                 | \$5        |
| ▼ Brazil                                                               | 36               | \$66       | 212               | \$224      | 145                | \$26       |
| Campinas                                                               |                  |            | 10                | \$34       |                    |            |
| Row pager < > Rows per page 10 ▼ Column pager < > Columns per page 5 ▼ |                  |            |                   |            |                    |            |

Show or hide paging option

By using the [ShowRowPager](#) and [ShowColumnPager](#) properties in [PagerSettings](#), you can show or hide row and column pager separately in the pager UI.

In the following example, row pager has been disabled by setting the [ShowRowPager](#) property in [PagerSettings](#) to **false**.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("ProductName").Caption("Product Name").Add();
}).Values(values =>
{
```



```

 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
 })
).PageSettings(pageSettings =>
pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings => pagerSettings.ShowRowPager(false)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()

```

## PAGING.CS

```

public ActionResult Index()
{
 return View();
}

```

|                                                                                                                           | Boston Crab Meat |            | Camembert Pierrot |            | Guaraná Fantástica |            |
|---------------------------------------------------------------------------------------------------------------------------|------------------|------------|-------------------|------------|--------------------|------------|
|                                                                                                                           | Quantity         | Unit Price | Quantity          | Unit Price | Quantity           | Unit Price |
| ▼ Argentina                                                                                                               | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| Buenos Aires                                                                                                              | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| ▼ Austria                                                                                                                 | 91               | \$18       | 160               | \$68       | 283                | \$22       |
| Graz                                                                                                                      | 91               | \$18       | 160               | \$68       | 248                | \$17       |
| Salzburg                                                                                                                  |                  |            |                   |            | 35                 | \$5        |
| ▼ Belgium                                                                                                                 | 15               | \$37       | 40                | \$27       | 12                 | \$5        |
| Bruxelles                                                                                                                 | 10               | \$18       |                   |            |                    |            |
| Charleroi                                                                                                                 | 5                | \$18       | 40                | \$27       | 12                 | \$5        |
| ▼ Brazil                                                                                                                  | 36               | \$66       | 212               | \$224      | 145                | \$26       |
| Campinas                                                                                                                  |                  |            | 10                | \$34       |                    |            |
| <div> <div>  &lt; &lt; &gt; &gt;  </div> <div>1 of 4</div> <div>Column pager</div> <div>Columns per page 5 ▼</div> </div> |                  |            |                   |            |                    |            |

### Show or hide page size

By using the [ShowRowPageSize](#) and [ShowColumnPageSize](#) properties in [PagerSettings](#), you can show or hide "Rows per page" and "Columns per page" dropdown menu. The dropdown menu contains a list of pre-defined or user-defined page sizes, which will be displayed in the "Rows per page" and "Columns per page" dropdowns, allowing you to change the page size for the row and column axes at runtime.

## CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}

```

```

 }).Rows(rows =>
 {
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
 }).Columns(columns =>
 {
 columns.Name("ProductName").Caption("Product Name").Add();
 }).Values(values =>
 {
 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
 })
).PageSettings(pageSettings =>
 pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings =>
 pagerSettings.ShowRowPageSize(false).ShowColumnPageSize(false)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()

```

### PAGING.CS

```

public ActionResult Index()
{
 return View();
}

```

|                 | Boston Crab Meat |            | Camembert Pierrot |            | Guaraná Fantástica |            |
|-----------------|------------------|------------|-------------------|------------|--------------------|------------|
|                 | Quantity         | Unit Price | Quantity          | Unit Price | Quantity           | Unit Price |
| ▼ Argentina     | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| Buenos Aires    | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| ▼ Austria       | 91               | \$18       | 160               | \$68       | 283                | \$22       |
| Graz            | 91               | \$18       | 160               | \$68       | 248                | \$17       |
| Salzburg        |                  |            |                   |            | 35                 | \$5        |
| ▼ Belgium       | 15               | \$37       | 40                | \$27       | 12                 | \$5        |
| Bruxelles       | 10               | \$18       |                   |            |                    |            |
| Charleroi       | 5                | \$18       | 40                | \$27       | 12                 | \$5        |
| ▼ Brazil        | 36               | \$66       | 212               | \$224      | 145                | \$26       |
| Campinas        |                  |            | 10                | \$34       |                    |            |
| Row pager       |                  |            | Column pager      |            |                    |            |
| < < > >  1 of 9 |                  |            | < < > >  1 of 4   |            |                    |            |

#### Customize page size

By using the [RowPageSizes](#) and [ColumnPageSizes](#) properties in [PagerSettings](#), you can specify a set of desired page sizes, which will be displayed in the "Rows per page" and "Columns per page" dropdowns, allowing you to change the page size for the row and column axes at runtime.

**Note:** By default, the "Rows per page" dropdown have pre-defined page sizes of **10, 50, 100, and 200**, while the "Columns per page" dropdown have pre-defined page sizes of **5, 10, 20, 50, and 100**.

In the following example, the "Rows per page" dropdown is set with user-defined page sizes of **10, 20, 30, 40, and 50** and the "Columns per page" dropdown is set with user-defined page sizes of **5, 10, 15, 20, and 30**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("ProductName").Caption("Product Name").Add();
}).Values(values =>
{
 values.Name("Quantity").Caption("Quantity").Add();
values.Name("UnitPrice").Caption("Unit Price").Add();
})
).PageSettings(pageSettings =>
pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentRowPage(1)
).PagerSettings(pagerSettings =>
pagerSettings.RowPageSizes(ViewBag.RowPageSizes).ColumnPageSizes(ViewBag.ColumnPageSizes)
).GridSettings(gridSettings => gridSettings.ColumnWidth(120)).Render()
```

### PAGING.CS

```
public ActionResult Index()
{
 ViewBag.RowPageSizes = new double[] { 10, 20, 30, 40, 50 };
 ViewBag.ColumnPageSizes = new double[] { 5, 10, 15, 20, 30 };
 return View();
}
```

|              | Boston Crab Meat |            | Camembert Pierrot |            | Guaraná Fantástica |            |
|--------------|------------------|------------|-------------------|------------|--------------------|------------|
|              | Quantity         | Unit Price | Quantity          | Unit Price | Quantity           | Unit Price |
| ▼ Argentina  | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| Buenos Aires | 20               | \$15       | 11                | \$19       | 36                 | \$35       |
| ▼ Austria    | 91               | \$18       | 160               | \$68       | 283                | \$22       |
| Graz         | 91               | \$18       | 160               | \$68       | 248                | \$17       |
| Salzburg     |                  |            |                   |            | 35                 | \$5        |
| ▼ Belgium    | 15               | \$37       | 40                | \$27       | 12                 | \$5        |
| Bruxelles    | 10               | \$18       |                   |            |                    |            |
| Charleroi    | 5                | \$18       | 40                | \$27       | 12                 | \$5        |
| ▼ Brazil     | 36               | \$66       | 212               | \$224      | 145                | \$26       |
| Campinas     |                  |            | 10                | \$34       |                    |            |

Row pager

1
of 9

Rows per page

10

Column pager

1
of 4

Columns per page

5

### Template

The [Template](#) property allows to change the appearance of the pager UI by displaying user-defined HTML elements instead of built-in HTML elements.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("350").EnablePaging(true).DataSourceSettings(dataSource => dataSource.DataSource(dataManger =>
{
 dataManger.Url("https://bi.syncfusion.com/northwindservice/api/orders").CrossDomain(true).Adaptor("WebApiAdaptor");
}).ExpandAll(true).ShowAggregationOnValueField(false).EnableSorting(true).FormatSettings(formatsettings =>
{
 formatsettings.Name("UnitPrice").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("ShipCountry").Add(); rows.Name("ShipCity").Add();
}).Columns(columns =>
{
 columns.Name("ProductName").Caption("Product Name").Add();
}).Values(values =>
{
 values.Name("Quantity").Caption("Quantity").Add();
 values.Name("UnitPrice").Caption("Unit Price").Add();
}))
```

```

).PageSettings(pageSettings =>
pageSettings.ColumnPageSize(5).RowPageSize(10).CurrentColumnPage(1).CurrentR
owPage(1)
).PagerSettings(pagerSettings => pagerSettings.Template("#template")
).GridSettings(gridSettings =>
gridSettings.ColumnWidth(120)).DataBound("onDataBound").Render()
<script id="template" type="text/x-template">
 <div class="pager-label">Row Pager: </div>
 <div id="row-pager" class="e-pagertemplate"></div>
 <div class="pager-label">Column Pager: </div>
 <div id="column-pager" class="e-pagertemplate"></div>
</script>
<script>
 function onDataBound() {
 updateTemplate();
 }
 function updateTemplate() {
 var pivotObj =
document.getElementById('PivotView').ej2_instances[0];
 if (!ej.base.isNullOrUndefined(rowPager)) {
 rowPager.destroy();
 rowPager = null;
 }
 rowPager = new ej.grids.Pager({
 pageSize: pivotObj.pageSettings.rowPageSize,
 totalRecordsCount: pivotObj.engineModule.rowCount,
 currentPage: pivotObj.pageSettings.currentRowPage,
 pageCount: 5,
 click: rowPageClick
 });
 rowPager.appendTo('#row-pager');
 if (!ej.base.isNullOrUndefined(columnPager)) {
 columnPager.destroy();
 columnPager = null;
 }
 columnPager = new ej.grids.Pager({
 pageSize: pivotObj.pageSettings.columnPageSize,
 totalRecordsCount: pivotObj.engineModule.columnCount,
 currentPage: pivotObj.pageSettings.currentColumnPage,
 pageCount: 5,
 click: columnPageClick
 });
 columnPager.appendTo('#column-pager');
 }
 function rowPageClick(args) {
 var pivotObj =
document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pageSettings.currentRowPage = args.currentPage;
 pivotObj.refreshData();
 }
 function columnPageClick(args) {
 var pivotObj =
document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pageSettings.currentColumnPage = args.currentPage;
 pivotObj.refreshData();
 }
</script>

```

```
<style>
.e-pivot-pager {
 display: flex;
}
.pager-label {
 color: #9e9e9e;
 margin-right: 10px;
}
#row-pager {
 margin-right: 10px;
}
</style>
```

PAGING.CS

```
public ActionResult Index()
{
 return View();
}
```

|              | Boston Crab Meat |            | Camembert Pierrot |            | Grand Total |
|--------------|------------------|------------|-------------------|------------|-------------|
|              | Quantity         | Unit Price | Quantity          | Unit Price |             |
| ▼ Argentina  | 20               | \$15       | 11                | \$19       |             |
| Buenos Aires | 20               | \$15       | 11                | \$19       |             |
| ▼ Austria    | 91               | \$18       | 160               | \$68       |             |
| Graz         | 91               | \$18       | 160               | \$68       |             |
| Salzburg     |                  |            |                   |            |             |
| ▼ Belgium    | 15               | \$37       | 40                | \$27       |             |
| Bruxelles    | 10               | \$18       |                   |            |             |
| Charleroi    | 5                | \$18       | 40                | \$27       |             |
| ▼ Brazil     | 36               | \$66       | 212               | \$224      |             |
| Campinas     |                  |            | 10                | \$34       |             |

Row Pager:

<<

<

1

2

3

4

5

...

>

>>

1 of 9 pages (88 items)

Column Pager:

<<

<

1

2

3

4

>

>>

1 of 4 pages (18 items)

State Persistence

State persistence allows user to maintain the current state of the component along with its report bounded in the browser local storage (cookie). Even if the browser is refreshed or if you move to the

next page within the browser, components state will be persisted. State persistence stores the Pivot Table object in the local storage when [EnablePersistence](#) property in [PivotView](#) class is set to **true**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).EnablePersistence(true).ShowGroupingBar(true).Render()
```

### PERSISTENCE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Save and Load Pivot Layout

You can save the current layout of the pivot table by using `getPersistData` in string format. The saved layout can be loaded to pivot table any time by passing the saved data as a parameter to `loadPersistData` method in the [PivotView](#).

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("save").Content("Save Layout").IsPrimary(true).Render()
@Html.EJS().Button("load").Content("Load Layout").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).EnablePersistence(true).ShowGroupingBar(true).Render()
```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowGroupingBar(true).ShowFieldList(true).Render()
<script>
 var pivotObj;
 var pivotLayout;
 document.getElementById('save').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotLayout = pivotObj.getPersistData();
 }
 document.getElementById('load').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.loadPersistData(pivotLayout);
 }
</script>

```

### SAVE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

<!-- markdownlint-disable MD012 -->

## Row and Column in ASP.NET MVC Pivot Table Component

### Width and Height

Allows end user to set the pivot table's height and width by using [Height](#) and [Width](#) properties in [PivotView](#) class respectively. The supported formats to set [Height](#) and [Width](#) properties are,

- Pixel: For example - 100, 200, "100px", "200px".
- Percentage: For example - "100%", "200%".
- Auto: It is applicable for [Height](#) property alone in-order to render the pivot table beyond its parent container height without vertical scrollbar. The parent container here would show its vertical scrollbar as soon as the component reaches beyond its dimension.

**Note:** The pivot table will not be displayed less than **400px**, since it's the minimum width of the component.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Width("550").Height("315px").DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 })
 }).Rows(rows =>

```



```
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).Render()
```

### SIZE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                 | ▼ FY 2015  |             |            |
|-----------------|------------|-------------|------------|
|                 | Q1         |             | Q2         |
|                 | Units Sold | Sold Amount | Units Sold |
| ▼ France        | 114        | \$177,246   | 108        |
| Mountain Bikes  | 31         | \$52,824    | 51         |
| Road Bikes      | 83         | \$124,422   | 57         |
| ▼ Germany       | 74         | \$117,246   | 128        |
| Mountain Bikes  | 51         | \$92,824    | 61         |
| Road Bikes      | 23         | \$24,422    | 67         |
| ▼ United States | 144        | \$162,246   | 171        |

### Row Height

Allows end user to set the height of each pivot table rows commonly using the [RowHeight](#) property in [PivotViewGridSettings](#) class.

**Note:** By default, the [RowHeight](#) property is set as **36** pixels for desktop layout and **48** pixels for mobile layout.

<br/> The height of the column headers alone may vary when grouping bar feature is enabled.

In the below code sample, the [RowHeight](#) property is set as **60** pixels.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(new PivotViewGridSettings { RowHeight = 60 }).Render()

```

### ROWHEIGHT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                    | FY 2015     |                    | FY 2016     |                    | FY 2017    |
|--------------------|-------------|--------------------|-------------|--------------------|------------|
|                    | Units Sold  | Sold Amount        | Units Sold  | Sold Amount        | Units Sold |
| France             | 450         | \$714,955          | 526         | \$1,542,104        |            |
| Germany            | 440         | \$563,515          | 496         | \$1,772,104        |            |
| United States      | 546         | \$754,515          | 636         | \$2,263,104        |            |
| <b>Grand Total</b> | <b>1436</b> | <b>\$2,032,585</b> | <b>1658</b> | <b>\$5,577,312</b> |            |

### Column Width

Allows end user to set the width of each pivot table columns commonly using the [ColumnWidth](#) property in [PivotViewGridSettings](#) class.

**Note:** By default, the [ColumnWidth](#) property is set as **110** pixels to each columns except the first column. For first column, **250** pixels and **200** pixels are set respectively with and without grouping bar.

In the below example, the [ColumnWidth](#) property is set as **200** pixels.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).GridSettings(new PivotViewGridSettings { ColumnWidth=120 })).Render()
```

### COLUMNWIDTH.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             |       |
|---------------|------------|-------------|------------|-------------|-------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |       |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |       |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |       |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |       |

*Adjust width based on columns*

By default, if the component width set in code-behind is more than the width of the total columns, then the columns will be stretched to make it fit. To avoid the stretching, set the [AllowAutoResizing](#) property in the [PivotViewGridSettings](#) to **false**. By doing so, the component will be adjusted (shrunk) based on the width of total columns.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FilterSettings(filterSettings =>
{
filterSettings.Name("Year").Type(Syncfusion.EJ2.PivotView.FilterType.Exclude)
.Items(ViewBag.filterMembers).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
})).GridSettings(new PivotViewGridSettings { AllowAutoResizing=true
}).Render()
```

**ALLOWAUTORESIZING.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.filterMembers = new string[] { "FY 2017", "FY 2015" };
 return View();
}
```

|                 | ► FY 2016 | ► FY 2018 | Grand Total |
|-----------------|-----------|-----------|-------------|
| ► France        | 526       | 16        | 542         |
| ► Germany       | 496       | 96        | 592         |
| ► United States | 636       | 76        | 712         |
| Grand Total     | 1658      | 188       | 1846        |

## Reorder

Allows end user to reorder a particular column header from one index to another index within the pivot table through drag-and-drop option. It can be enabled by setting the [AllowReordering](#) property in [PivotViewGridSettings](#) class to **true**.

## CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(new PivotViewGridSettings { AllowReordering=true
}).Render()
```

## REORDER.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                 | ► FY 2015  | FY 2016     |            | ► FY 2017   |            |
|-----------------|------------|-------------|------------|-------------|------------|
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ► France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| ► Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| ► United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total     | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Column Resizing

Allows end user to resize the columns by clicking and dragging the right edge of the column header. While dragging, the width of the respective column will be resized immediately. To enable column resizing option, set the [AllowResizing](#) property in [PivotViewGridSettings](#) class to **true**.

**Note:** By default, the column resizing option is enabled.

<br/> In RTL mode, user can click and drag the left edge of the header cell to resize the column.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).GridSettings(new PivotViewGridSettings { AllowResizing=true }).Render()
```

### COLUMNRESIZING.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 5          |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 3          |
| United States | 546        | \$754,515   | 636        | \$2,263,104 | 7          |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 16         |

### Text Wrap

Allows end user to wrap the cell content to the next line when it exceeds the boundary of the cell width. To enable text wrap, set the [AllowTextWrap](#) property in [PivotViewGridSettings](#) class to **true**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).GridSettings(new PivotViewGridSettings { AllowTextWrap=true }).Render()
```

### TEXTWRAP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |             |
|---------------|------------|-------------|------------|-------------|------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 592        | \$2,903,308 |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 372        | \$1,634,808 |
| United States | 546        | \$754,515   | 636        | \$2,263,104 | 731        | \$3,387,308 |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1695       | \$7,925,424 |

### Text Align

Allows end user to align the content of the pivot table's row and column headers and value cells by using the [TextAlign](#) and [HeaderTextAlign](#) properties in the [ColumnRender](#) event under [GridSettings](#). The following alignments are:

- **Left** - It allows the content to be positioned on the left.
- **Right** - It allows the content to be positioned on the right.
- **Center** - It allows the content to be positioned in the middle.
- **Justify** - It allows the content to be as flexible as possible, when the cell does not occupy the entire available area.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).GridSettings(gridSettings =>
gridSettings.ColumnRender("columnRender")).Render()
<script>
function columnRender(args) {
 if(args.stackedColumns[0]){
 // Content for the row headers is right-aligned here.
 args.stackedColumns[0].textAlign="Right";
 }
 if(args.stackedColumns[1]){
 // Content for the column header "FY 2015" is center-aligned
 here.
 args.stackedColumns[1].textAlign = 'Center';
 }
 if(args.stackedColumns[1] && args.stackedColumns[1].columns[0]){
 // Content for the column header "Q1" is right-aligned here.
 args.stackedColumns[1].columns[0].textAlign = 'Right';
 }
 if(args.stackedColumns[1] && args.stackedColumns[1].columns[0] &&
args.stackedColumns[1].columns[0].columns[0]){
 // Content for the value header "Units Sold" is right-aligned
 here.
 args.stackedColumns[1].columns[0].columns[0].headerTextAlign =
'Right';
 }
 if(args.stackedColumns[1] && args.stackedColumns[1].columns[0] &&
args.stackedColumns[1].columns[0].columns[0]){
```



```
// Content for the values are left-aligned here.
args.stackedColumns[1].columns[0].columns[0].textAlign = 'Left';
 }
}
</script>
```

### TEXTWRAP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                  | FY 2015    |             |            |             |            |             |            |
|------------------|------------|-------------|------------|-------------|------------|-------------|------------|
|                  | Q1         |             | Q2         |             | Q3         |             | Q4         |
|                  | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ▼ France         | 203        | \$319,246   | 199        | \$317,543   | 134        | \$221,637   | 193        |
| Mountain Bikes   | 31         | \$52,824    | 51         | \$86,904    | 90         | \$153,360   | 25         |
| Road Bikes       | 83         | \$124,422   | 57         | \$85,448    | 20         | \$29,985    | 93         |
| Touring Bikes    | 89         | \$142,000   | 91         | \$145,191   | 24         | \$38,292    | 75         |
| ▼ Germany        | 111        | \$183,591   | 127        | \$201,679   | 143        | \$231,933   | 147        |
| Mountain Bikes   | 74         | \$126,096   | 33         | \$56,232    | 65         | \$110,760   | 16         |
| Road Bikes       | 16         | \$23,989    | 47         | \$70,458    | 34         | \$50,971    | 83         |
| Touring Bikes    | 21         | \$33,506    | 47         | \$74,989    | 44         | \$70,202    | 48         |
| ▼ United Kingdom | 183        | \$298,020   | 183        | \$298,972   | 136        | \$218,667   | 280        |
| Mountain Bikes   | 77         | \$131,208   | 92         | \$156,768   | 51         | \$86,904    | 91         |

### AutoFit

Allows the user to fit the Pivot Table columns as wide as the content of the cell without wrapping. It auto fits all of the Pivot Table columns by invoking the [autoFitColumns](#) method from the grid instance.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false);
 dataSourceSettings.FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 });
 });
```

```

 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DataBound("ondataBound").Render()
<script>
 function ondataBound(args) {
 var pivotTableObj =
 document.getElementById('PivotView').ej2_instances[0];
 pivotTableObj.grid.autoFitColumns();
 }
</script>

```

### AUTOFITMETHOD.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 59         |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 37         |
| United Sta... | 546        | \$754,515   | 636        | \$2,263,104 | 63         |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 159        |

**Note:** The minimum width of 250 pixels is set by default with the grouping bar UI for the first column and cannot be reduced further. So, when the grouping bar is enabled, one can auto fit the Pivot Table columns by calling the [autoFitColumns](#) method from the grid instance with the parameter contained pivot table columns field name excluding first column.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 })
 }).Rows(rows =>
 {

```

```

 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DataBound("ondataBound").ShowGroupingBar(true).Render()
<script>
 function ondataBound(args) {
 var pivotTableObj =
document.getElementById('PivotView').ej2_instances[0];
 var columns = [];
 for (var i = 1; i < pivotTableObj.grid.columnModel.length; i++) {
 columns.push(pivotTableObj.grid.columnModel[i].field);
 }
 pivotTableObj.grid.autoFitColumns(columns);
 }
</script>

```

### AUTOFIT-GROUPINGBAR.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                    |                  |             |            |             |
|--------------------|------------------|-------------|------------|-------------|
| Sum of Units Sold  | Drop filter here |             |            |             |
| Sum of Sold Amount | Year Quarter     |             |            |             |
| Country            | FY 2015          |             | FY 2016    |             |
| Products           | Units Sold       | Sold Amount | Units Sold | Sold Amount |
| France             | 450              | \$714,955   | 526        | \$1,54      |
| Germany            | 440              | \$563,515   | 496        | \$1,77      |
| United States      | 546              | \$754,515   | 636        | \$2,26      |
| Grand Total        | 1436             | \$2,032,985 | 1658       | \$5,57      |

### Autofit specific columns

During initial rendering, the parameter `autoFit` in the [ColumnRender](#) event under [PivotViewGridSettings](#) can be set to **true** to auto fit specific columns.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.ColumnRender("columnRender")).Render()
<script>
function columnRender(args) {
 for (var i = 0; i < args.columns.length; i++) {
 args.columns[i].autoFit = true;
 }
}
</script>

```

### AUTOFIT-EVENT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                | ► FY 2015  |             | ► FY 2016  |             | ► FY 2017  |
|----------------|------------|-------------|------------|-------------|------------|
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| ► France       | 450        | \$714,955   | 526        | \$1,542,104 | 592        |
| ► Germany      | 440        | \$563,515   | 496        | \$1,772,104 | 372        |
| ► United St... | 546        | \$754,515   | 636        | \$2,263,104 | 632        |
| Grand Total    | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1596       |

### Grid Lines

Allows end user to display cell border for each cells using [GridLines](#) property in [PivotViewGridSettings](#) class.

Available mode of grid lines are:

- | Modes | Actions |
- |-----|-----|
- | Both | Displays both the horizontal and vertical grid lines.|
- | None | No grid lines are displayed.|
- | Horizontal | Displays the horizontal grid lines only.|
- | Vertical | Displays the vertical grid lines only.|
- | Default | Displays grid lines based on the theme.|

**Note:** By default, pivot table renders grid lines in **Both** mode.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(new PivotViewGridSettings { GridLines= "Vertical"
}).Render()
```

### GRIDLINES.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Selection

Selection provides an option to highlight a row or a column or a cell. It can be done through simple mouse down or arrow keys. To enable selection in the pivot table, set the [AllowSelection](#) property in [PivotViewGridSettings](#) class to **true**.

The pivot table supports two types of selection that can be set using [Type](#) property in [PivotViewSelectionSettings](#) class. The selection types are:

- **Single:** It is set by default, and it only allows selection of a single row or a column or a cell.
- **Multiple:** Allows you to select multiple rows or columns or cells.

To perform multi-selection, press and hold "CTRL" key and click the desired rows or cells. To select range of rows or cells, press and hold the "SHIFT" key and click the rows or columns or cells.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.AllowSelection(true).SelectionSettings(selectionSettings =>
selectionSettings.Type("Multiple"))).Render()
```

### SELECTION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Selection Mode

The pivot table supports four types of selection mode that can be set using [Mode](#) property in [PivotViewSelectionSettings](#) class.. The selection modes are:

- [SelectionMode.Row](#): It is set by default, and allows user to select only rows.
- [SelectionMode.Column](#): Allows user to select only columns.
- [SelectionMode.Cell](#): Allows user to select only cells.
- [SelectionMode.Both](#): Allows user to select rows and columns at the same time.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).GridSettings(gridSettings =>
 gridSettings.AllowSelection(true).SelectionSettings(selectionSettings =>
 selectionSettings.Mode(SelectionMode.Both)).Render()
```

**SELECTIONMODE.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

*Cell Selection Mode*

The pivot table supports two types of cell selection mode that can be set using [CellSelectionMode](#) in [PivotViewSelectionSettings](#) class. The cell selection modes are:

- **[PivotCellSelectionMode.Flow](#)**: It is set by default. The range of cells are selected between the start index and end index that includes in-between cells of rows.
- **[PivotCellSelectionMode.Box](#)**: Range of cells are selected from the start and end column indexes that includes in-between cells of rows within the range.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
{
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
})
```



```

))) .GridSettings(gridSettings =>
gridSettings.AllowSelection(true) .SelectionSettings(selectionSettings =>
selectionSettings.CellSelectionMode("Box") .Type("Multiple") .Mode(SelectionMode.Both)) .Render()

```

### CELLSELECTION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

**Note:** Cell selection requires [Mode](#) property in [PivotViewSelectionSettings](#) class to be [SelectionMode.Cell](#) or [SelectionMode.Both](#), and [Type](#) property should be [Multiple](#).

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

#### Changing background color of the selected cell

The background-color of the selected cell can be changed using built-in CSS names. To do so, refer to the code sample below, which shows that the selected cells are changed to a **green yellow** color.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{

```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).GridSettings(gridSettings =>
 gridSettings.AllowSelection(true).SelectionSettings(selectionSettings =>
 selectionSettings.CellSelectionMode("Box").Type("Multiple").Mode(SelectionMode.Both))).Render()
<style>
 .e-pivotview .e-cellselectionbackground,
 .e-pivotview .e-selectionbackground,
 .e-pivotview .e-grid .e-rowsheader.e-selectionbackground,
 .e-pivotview .e-grid .e-columnsheader.e-selectionbackground {
 background-color: greenYellow !important;
 }
</style>

```

### CELLSELECTION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                  | ► FY 2015  |             | ► FY 2016  |             |
|------------------|------------|-------------|------------|-------------|
|                  | Units Sold | Sold Amount | Units Sold | Sold Amount |
| ▼ France         | 729        | \$1,160,100 | 609        |             |
| Mountain Bikes   | 197        | \$335,688   | 238        |             |
| Road Bikes       | 253        | \$379,267   | 147        |             |
| Touring Bikes    | 279        | \$445,145   | 224        |             |
| ► Germany        | 528        | \$845,472   | 667        | \$          |
| ► United Kingdom | 782        | \$1,263,110 | 640        | \$          |
| ► United States  | 682        | \$1,085,399 | 480        |             |

### Event

#### CellSelected

The event **CellSelected** is triggered when cell selection gets completed. It provides selected cells information with its corresponding column and row headers. It has following parameters -

`selectedCellsInfo`, `currentCell` and `target`. This event allows user to view selected cells information and user can pass those selected cells information to any external component for data binding.

### CSHTML

```
<div>
<div class="column-8">
 @using Syncfusion.EJ2.PivotView

 @Html.EJS().PivotView("pivotview").Width("100%").Height("400").DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)
 ViewBag.DataSource).ExpandAll(true).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product Categories").Add();
 })
).GridSettings(gridSettings =>
 {
 gridSettings.AllowSelection(true).SelectionSettings(selectionSettings =>
 {
 selectionSettings.CellSelectionMode(PivotCellSelectionMode.Box).Type("Multiple").Mode(SelectionMode.Cell)).CellSelected("onCellSelected").Render()
 })
 })
 })
 </div>
 <div class="column-3">
 <div class="eventarea" style="height: 230px;overflow: auto">

 </div>
 </div>
</div>
<script>
 function onCellSelected(args) {
 document.getElementById('selection-EventLog').innerHTML = '';
 if (args.selectedCellsInfo.length > 0) {
 for (var cnt = 0; cnt < args.selectedCellsInfo.length; cnt++) {
 var cell = args.selectedCellsInfo[cnt];
 }
 }
 }
</script>
```

```

 var summMeasure = this.engineModule.fieldList[cell.measure]
? this.engineModule.fieldList[cell.measure].aggregateType + ' of ' +
 this.engineModule.fieldList[cell.measure].caption : '';
 appendElement(
 (cell.columnHeaders == '' ? '' : 'Column Headers: ' +
'' + cell.columnHeaders.split('.').join(' - ') + '</br>') +
 (cell.rowHeaders == '' ? '' : 'Row Headers: ' + '' +
cell.rowHeaders.split('.').join(' - ') + '</br>') +
 (summMeasure == '' ? '' : 'Measure: ' + '' +
summMeasure + '</br>') +
 'Value: ' + '' + cell.currentCell.formattedText +
'<hr></br>');
 }
 }
}
function appendElement(html) {
 var span = document.createElement('span');
 span.innerHTML = html;
 var log = document.getElementById('selection-EventLog');
 log.appendChild(span);
}
var modeddl = new ej.dropdowns.DropDownList({
 floatLabelType: 'Auto',
 width: 150,
 change: function (args) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.gridSettings.selectionSettings.mode = args.value;
 pivotObj.renderModule.updateGridSettings();
 }
});
modeddl.appendTo('#mode');
var typeddl = new ej.dropdowns.DropDownList({
 floatLabelType: 'Auto',
 width: 150,
 change: function (args) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.gridSettings.selectionSettings.type = args.value;
 pivotObj.renderModule.updateGridSettings();
 }
});
typeddl.appendTo('#type');
</script>

```

### CELLSELECTION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             |
|---------------|------------|-------------|------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 |

Event Trace:

Column Headers: FY 2015

Row Headers: France

Value: 450

Measure: Sold

Column Headers: FY 2015

Row Headers: Germany

Value: 440

Measure: Sold

### CellSelecting

The event **CellSelecting** triggers before cell gets selected gets completed. It provides selected cells information with its corresponding column and row headers. It has following parameters - **currentCell**, **data** and **cancel**.

### CSHTML

```
<div>
<div class="column-8">
 @using Syncfusion.EJ2.PivotView

 @Html.EJS().PivotView("pivotview").Width("100%").Height("400").DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)
 ViewBag.DataSource).ExpandAll(true).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 columns.Name("Order_Source").Caption("Order Source").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 })
 .Filters(filters =>
 {
 filters.Name("Product_Categories").Caption("Product Categories").Add();
 })
).GridSettings(gridSettings =>
 {
 gridSettings.AllowSelection(true).SelectionSettings(selectionSettings =>
 {
 selectionSettings.CellSelectionMode(PivotCellSelectionMode.Box).Type("Multiple").Mode(SelectionMode.Cell)
 }).CellSelecting("cellSelecting").Render()
 })
 })
 </div>
</script>
```

```
function cellSelecting(args) {
 }
</script>
```

### CELLSELECTION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Clip Mode

The clip mode provides options to display its overflow cell content in the pivot table. It can be configured using the [ClipMode](#) property in [PivotViewGridSettings](#) class. The pivot table supports three types of clip modes which are:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

**Note:** By default, [ClipMode](#) value is set to **Ellipsis**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings => gridSettings.ClipMode("Clip")).Render()
```

### CLIPMODE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015 |          | FY 2016    |             | FY 2017    |             |
|---------------|---------|----------|------------|-------------|------------|-------------|
|               | Units   | Sold     | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450     | \$714,95 | 526        | \$1,542,104 | 592        | \$2,903,    |
| Germany       | 440     | \$563,51 | 496        | \$1,772,104 | 372        | \$1,634,    |
| United States | 546     | \$754,51 | 636        | \$2,263,104 | 632        | \$3,041,    |
| Grand Total   | 1436    | \$2,032, | 1658       | \$5,577,312 | 1596       | \$7,579,    |

### Cell Template

User can customize the pivot table cell element by using the `CellTemplate` property in `PivotViewTemplates` class. The `CellTemplate` property accepts either an HTML string or the element's ID, which can be used to append additional HTML elements to showcase each cell with custom format.

In this demo, the revenue cost for each year is represented with trend icons.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").DataBound("trend").DataSourceSettings(dataSourceSettings =>
{
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(true).EnableSorting(true)
 .DrilledMembers(drilledmembers =>{
 drilledmembers.Name("Year").Items(ViewBag.drilledMembers).Add();
 })
 .FormatSettings(formatsettings => {
 formatsettings.Name("ProCost").Format("C0").Add();
 })
 .Rows(rows => {
 rows.Name("Year").Caption("Year").Add();
 rows.Name("HalfYear").Caption("Half Year").Add();
 })
 .Columns(columns => {
 columns.Name("EnerType").Caption("Energy Type").Add();
 columns.Name("EneSource").Caption("Energy Source").Add();
 })
 .Values(values => {
 values.Name("ProCost").Caption("Revenue Growth").Add();
 })
}).GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).CellTemplate("${getCellContent(data)}").Render()

<script>
 window.getCellContent = function (e) {
 var template;
 if (e && e.targetCell.className.indexOf('e-valuescontent') > -1)
 {
```

```

 template = '<span class="tempwrap sb-icon-neutral pv-
icons">';
 } else {
 template = '';
 }
 return template;
};
/* jshint ignore:start */
function trend() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 var cTable = document.getElementsByClassName("e-table");
 var colLen = pivotObj.pivotValues[3].length;
 var cLen = cTable[3].children[0].children.length;
 var rLen = cTable[3].children[1].children.length;
 for (let k = 0; k < rLen; k++) {
 if (pivotObj.pivotValues[k] && pivotObj.pivotValues[k][0]
!= undefined) {
 rowIndx = (pivotObj.pivotValues[k][0]).rowIndex;
 break;
 }
 }
 var rowHeaders =
[[]].slice.call(cTable[2].children[1].querySelectorAll('td'));
 var rows = pivotObj.dataSourceSettings.rows;
 if (rowHeaders.length > 1) {
 for (var i = 0, Cnt = rows; i < Cnt.length; i++) {
 var fields = {};
 var fieldHeaders = [];
 for (var j = 0, Lnt = rowHeaders; j < Lnt.length; j++) {
 var header = rowHeaders[j];
 if (header.className.indexOf('e-gtot') === -1 &&
header.className.indexOf('e-rowsheader') > -1 &&
header.getAttribute('fieldname') === rows[i].name) {
 var headerName =
rowHeaders[j].getAttribute('fieldname') + '_' + rowHeaders[j].textContent;
 fields[rowHeaders[j].textContent] = j;
 fieldHeaders.push(rowHeaders[j].textContent);
 }
 }
 if (i === 0) {
 for (var rnt = 0, Lnt = fieldHeaders; rnt <
Lnt.length; rnt++) {
 if (rnt !== 0) {
 var row = fields[fieldHeaders[rnt]];
 var prevRow = fields[fieldHeaders[rnt - 1]];
 for (var j = 0, ci = 1; j < cLen && ci <
colLen; j++ , ci++) {
 if
(!cTable[3].children[1].children[row]) {
 break;
 }
 var node =
cTable[3].children[1].children[row].childNodes[j];
 var prevNode =
cTable[3].children[1].children[prevRow].childNodes[j];
 var ri = undefined;

```



```

var prevRi = undefined;
if (node) {
 ri = node.getAttribute('index');
}
if (prevNode) {
 prevRi =
prevNode.getAttribute('index');
}
if (ri && ri <
pivotObj.pivotValues.length) {
 if
((pivotObj.pivotValues[prevRi][ci]).value >
(pivotObj.pivotValues[ri][ci]).value && node.querySelector('.tempwrap')) {
 var trendElement =
node.querySelector('.tempwrap');
 trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-loss');
 } else if
((pivotObj.pivotValues[prevRi][ci]).value <
(pivotObj.pivotValues[ri][ci]).value && node.querySelector('.tempwrap')) {
 var trendElement =
node.querySelector('.tempwrap');
 trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-profit');
 }
}
}
} else {
 for (var rnt = 0, Lnt = fieldHeaders; rnt <
Lnt.length; rnt++) {
 var row = fields[fieldHeaders[rnt]];
 for (var j = 0, ci = 1; j < cLen && ci < colLen;
j++ , ci++) {
 if (!cTable[3].children[1].children[row]) {
 break;
 }
 var node =
cTable[3].children[1].children[row].childNodes[j];
 var prevNode =
cTable[3].children[1].children[row - 1].childNodes[j];
 var ri = undefined;
 var prevRi = undefined;
 if (node) {
 ri = node.getAttribute('index');
 }
 if (prevNode) {
 prevRi = prevNode.getAttribute('index');
 }
 if (ri && ri < pivotObj.pivotValues.length)
{
 var cRowFieldName =
cTable[2].children[1].children[row].childNodes[0].getAttribute('fieldname');
 var prevRowFieldName =
cTable[2].children[1].children[row -
1].childNodes[0].getAttribute('fieldname');

```

```

 if
 ((pivotObj.pivotValues[prevRi][ci]).value >
 (pivotObj.pivotValues[ri][ci]).value && node.querySelector('.tempwrap') &&
 cRowFieldName === prevRowFieldName)
 {
 var trendElement =
 node.querySelector('.tempwrap');
 trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-loss');
 } else if
 ((pivotObj.pivotValues[prevRi][ci]).value <
 (pivotObj.pivotValues[ri][ci]).value && node.querySelector('.tempwrap') &&
 cRowFieldName === prevRowFieldName)
 {
 var trendElement =
 node.querySelector('.tempwrap');
 trendElement.className =
trendElement.className.replace('sb-icon-neutral', 'sb-icon-profit');
 }
 }
}
}
}
}
/* jshint ignore:end */
</script>

```

**CELL-TEMPLATE.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | ▼ Biomass  |   |              |   |         |   |       |   |     |
|---------------|------------|---|--------------|---|---------|---|-------|---|-----|
|               | Bio-diesel |   | Ethanol Fuel |   | Wastage |   | Wood  |   | Bi  |
| ▼ FY 2015     | \$438      |   | \$275        |   | \$252   |   | \$78  |   | \$1 |
| H1 FY 2015... | \$167      | → | \$183        | ↑ | \$91    | ↓ | \$53  | ↓ | \$4 |
| H2 FY 2015... | \$271      | ↑ | \$92         | ↓ | \$161   | → | \$25  | ↓ | \$5 |
| ▼ FY 2016     | \$344      |   | \$331        |   | \$335   |   | \$126 |   | \$1 |
| H1 FY 2016... | \$135      | → | \$173        | → | \$147   | → | \$76  | ↓ | \$5 |
| H2 FY 2016... | \$209      | ↑ | \$158        | → | \$188   | ↑ | \$50  | ↓ | \$6 |
| ▼ FY 2017     | \$565      |   | \$401        |   | \$367   |   | \$122 |   | \$1 |

## Events

### QueryCellInfo

The event `queryCellInfo` triggers while rendering each row and value cells in the pivot table. It allows the user to customize the current cell like adding or removing styles, editing value, etc. It has the following parameters:

- `cell` - It holds the current cell information.
- `data` - It holds the entire row data besides the current cell.
- `column` - It holds the entire column data besides the current cell.
- `pivotview` - It holds pivot table instance.

## CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
{
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
})
```

```

 })).GridSettings(gridSettings =>
 gridSettings.QueryCellInfo("querycell").Render()
<script>
 function queryCell(args) {
 }
</script>

```

### QUERYCELL.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### HeaderCellInfo

The event `headerCellInfo` triggers while rendering each column header cell in the pivot table. It allows the user to customize the element of the current header cell like adding or removing styles, editing value, etc. It has the following parameters:

- `node` - It holds the current header cell information.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.HeaderCellInfo("headerCellInfo").Render()
<script>
 function headerCellInfo(args) {
 }
</script>

```

**HEADERCELL.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*ColumnRender*

The event [ColumnRender](#) triggers while framing each columns for rendering in the pivot table. It allows the user to customize the text alignment, column visibility, autofit, re-ordering, minimum and maximum width for a specific column. It has the following parameters:

- **columns** - It holds the leaf level columns (i.e., value headers) information.
- **dataSourceSettings** - It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **name** - It holds the name of the event.
- **stackedColumns** - It holds the drilled columns (i.e., including column and value headers) information.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.ColumnRender("columnRender")).Render()
<script>
 function columnRender(args) {
 for (var i = 0; i < args.columns.length; i++) {
 args.columns[i].autoFit = true;
 args.columns[i].textAlign="Right";
 }
 }
</script>
```

**TEXTWRAP.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*CellClick*

The event **CellClick** triggers while clicking a cell in the pivot table. For instance, using this event end-user can either add or remove styles, edit value and also perform any other DOM manipulations. It has the following parameters:

- **currentCell** - It holds the current cell information.
- **data** - It holds the clicked cell's data like axis, formatted text, actual text, row header, column header and value informations.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingbar(true).CellClick("cellClick").Render()
<script>
 function cellClick(args){
 args.currentCell.setAttribute("style", "background-color: red;");
 }
</script>
```

**CELLCLICK.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

See Also

- [Show/hide tooltip](#)

## Show or hide totals in ASP.NET MVC Syncfusion Pivot Table Control

### Show or hide grand totals

Allows to show or hide grand totals in rows and columns using the [ShowGrandTotals](#) property. To hide the grand totals in rows and columns, set the property [ShowGrandTotals](#) in [PivotViewDataSourceSettings](#) class to **false**.

End user can also hide grand totals for row or columns separately by setting the property [ShowRowGrandTotals](#) or [ShowColumnGrandTotals](#) in [DataSourceSettings](#) class to **false** respectively.

By default, [ShowGrandTotals](#), [ShowRowGrandTotals](#) and [ShowColumnGrandTotals](#) properties in [PivotViewDataSourceSettings](#) class are set as **true**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
ShowRemoveIcon =false }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowGrandTotals(false)).Render()
```

### GRANDTOTAL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |

Show grand totals at top or bottom

Allows to show grand totals either at top or bottom in rows and columns using the [GrandTotalsPosition](#) property. To show the grand totals at top in rows and columns, set the [GrandTotalsPosition](#) property in [DataSourceSettings](#) to **Top**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
{
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 .GrandTotalsPosition(GrandTotalsPosition.Top).Render()
```

### GRANDTOTAL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



|                  | Grand Total |             | FY 2005    |
|------------------|-------------|-------------|------------|
|                  | Units Sold  | Sold Amount | Units Sold |
| Grand Total      | 15249       | 1397530     | 3          |
| ▶ Canada         | 3163        | 296040      |            |
| ▶ France         | 2680        | 287200      |            |
| ▶ Germany        | 3756        | 379920      |            |
| ▶ United Kingdom | 680         | 41760       |            |
| ▶ United States  | 4970        | 392610      | 1          |

### Show or hide sub-totals

Allows to show or hide sub-totals in rows and columns using the [ShowSubTotals](#) property. To hide all the sub-totals in rows and columns, set the property [ShowSubTotals](#) in [DataSourceSettings](#) class to **false**.

End user can also hide sub-totals for rows or columns separately by setting the property [ShowRowSubTotals](#) or [ShowColumnSubTotals](#) in [DataSourceSettings](#) class to **false** respectively.

By default, [ShowSubTotals](#), [ShowRowSubTotals](#) and [ShowColumnSubTotals](#) in [DataSourceSettings](#) class are set as **true**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).DrilledMembers(drilledmembers =>
{
drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowSubTotals(false)).Render()
```

**SUBTOTAL.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                | FY 2015    |             |            |             |       |
|----------------|------------|-------------|------------|-------------|-------|
|                | Q1         |             | Q2         |             | Q3    |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units |
| France         |            |             |            |             |       |
| Mountain Bikes | 31         | \$52,824    | 51         | \$86,904    |       |
| Road Bikes     | 83         | \$124,422   | 57         | \$85,448    |       |
| Germany        |            |             |            |             |       |
| Mountain Bikes | 51         | \$92,824    | 61         | \$76,904    |       |

**Show or hide sub-totals for specific fields**

Allows to show or hide sub-totals for specific fields in rows and columns using the [ShowSubTotals](#) property. To hide sub-totals for a specific field in row or column axis, set the property [ShowSubTotals](#) property in [Row](#) or [Column](#) class to **false** respectively.

By default, [ShowSubTotals](#) property in [Row](#) or [Column](#) class is set as **true**.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%).Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings
{
 ShowSortIcon = false
}).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").ShowSubTotals(false).Add();
 rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").ShowSubTotals(false).Caption("Year").Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 }
```

```
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .Render();
```

### SUBTOTALSPECIFIC.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                | FY 2015    |             |            |             |       |
|----------------|------------|-------------|------------|-------------|-------|
|                | Q1         |             | Q2         |             | Q3    |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units |
| France         |            |             |            |             |       |
| Mountain Bikes | 31         | \$52,824    | 51         | \$86,904    |       |
| Road Bikes     | 83         | \$124,422   | 57         | \$85,448    |       |
| Germany        |            |             |            |             |       |
| Mountain Bikes | 51         | \$92,824    | 61         | \$76,904    |       |

#### Show sub-totals at top or bottom

Allows to show sub-totals either at top or bottom of the header group in rows and columns by using the [SubTotalsPosition](#) property. By default, [SubTotalsPosition](#) property is set to **Auto**, which means that column sub-totals are displayed at the bottom and row sub-totals are displayed at the top of the header group in the pivot table.

To show sub-totals at top of the header group in rows and columns, set the [SubTotalsPosition](#) property in [DataSourceSettings](#) to **Top**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSettings(
 new PivotViewGroupingBarSettings {
 ShowRemoveIcon = false
 }).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10)
 .MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
```

```
columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).SubTotalsPosition(SubTotalsPosition.Top).Render()
```

### GRANDTOTAL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                   | FY 2015       |      |      | FY 2016 |
|-------------------|---------------|------|------|---------|
|                   | FY 2015 Total | Q1   | Q2   |         |
| ▼ France          | 5854          | 2766 | 3088 |         |
| Bottles and Cages | 1912          | 835  | 1077 |         |
| Cleaners          | 1788          | 817  | 971  |         |
| Fenders           | 2154          | 1114 | 1040 |         |
| ► Germany         | 5240          | 2796 | 2444 |         |
| ► United States   | 6568          | 3068 | 3500 |         |
| Grand Total       | 17662         | 8630 | 9032 | 1       |

To show sub-totals at bottom of the header group in rows and columns, set the [SubTotalsPosition](#) property in [DataSourceSettings](#) to **Bottom**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").GroupingBarSe
ttings(new PivotViewGroupingBarSettings {
ShowRemoveIcon =false }).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
```

```
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).SubTotalsPosition(SubTotalsPosition.Bottom).Render();
```

### GRANDTOTAL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                   | FY 2015 |      |               | FY 201 |
|-------------------|---------|------|---------------|--------|
|                   | Q1      | Q2   | FY 2015 Total |        |
| ▼ France          |         |      |               |        |
| Bottles and Cages | 835     | 1077 | 1912          |        |
| Cleaners          | 817     | 971  | 1788          |        |
| Fenders           | 1114    | 1040 | 2154          |        |
| France Total      | 2766    | 3088 | 5854          |        |
| ► Germany         | 2796    | 2444 | 5240          |        |
| ► United States   | 3068    | 3500 | 6568          |        |

### Show or hide totals using toolbar

It can also be achieved using built-in toolbar options by setting the [ShowToolbar](#) property in [PivotView](#) class to **true**. Also, include the items **GrandTotal** and **SubTotal** within the [Toolbar](#) property in [PivotView](#) class. End user can now see "Show/Hide Grand totals" and "Show/Hide Sub totals" icons in toolbar UI automatically.

The grand totals and sub-totals can be dynamically displayed at the top or bottom of the pivot table's row and column axes by using the built-in options "Grand totals position" and "Subtotals position" available in the grand totals and sub-totals drop down menus, respectively.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolbar(true).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 })
 .Columns(columns =>
```

```

{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).Toolbar(new List<string>() { "SubTotal", "GrandTotal" }).Render()

```

### TOOLBAR.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                |  | FY 2016     |            | FY 2017     |
|----------------|--|-------------|------------|-------------|
|                |  | Sold Amount | Units Sold | Sold Amount |
| France         |  | \$1,160,100 | 609        | \$983,317   |
| Mountain Bikes |  | \$335,688   | 238        | \$405,552   |
| Germany        |  | \$379,267   | 147        | \$220,373   |
| Mountain Bikes |  | \$445,145   | 224        | \$357,392   |
| Germany        |  | \$845,472   | 667        | \$1,067,220 |
| Mountain Bikes |  | \$320,352   | 226        | \$385,104   |

|                |  | FY 2016 |             | FY 2017    |
|----------------|--|---------|-------------|------------|
|                |  | Sold    | Sold Amount | Units Sold |
| Germany        |  | 729     | \$1,160,100 | 609        |
| Mountain Bikes |  | 197     | \$335,688   | 238        |
| Germany        |  | 253     | \$379,267   | 147        |
| Mountain Bikes |  | 279     | \$445,145   | 224        |
| Germany        |  | 528     | \$845,472   | 667        |
| Mountain Bikes |  | 188     | \$320,352   | 226        |

## Hyperlink

The pivot table supports to show hyperlink option to link data for individual cells that are displayed in the component. Also, the hyperlink can be enabled separately for row headers, column headers, value cells, and summary cells using the [HyperlinkSettings](#) class. It can be configured through code behind, during initial rendering and the settings available to show hyperlink are:

- [ShowHyperlink](#): It allows to set the visibility of hyperlink in all cells.
- [ShowRowHeaderHyperlink](#): It allows to set the visibility of hyperlink in row headers.
- [ShowColumnHeaderHyperlink](#): It allows to set the visibility of hyperlink in column headers.
- [ShowValueCellHyperlink](#): It allows to set the visibility of hyperlink in value cells.
- [ShowSummaryCellHyperlink](#): It allows to set the visibility of hyperlink in summary cells.
- [HeaderText](#): It allows to set the visibility of hyperlink based on header text.
- [ConditionalSettings](#): It allows to set the visibility of hyperlink based on specific condition.

<!-- markdownlint-disable MD028 -->

**Note:** By default, the hyperlink options are disabled for all cells in the pivot table.

**Note:** User defined style can be applied to hyperlink using [CssClass](#) property in [HyperlinkSettings](#) class.

### Hyperlink for all cells

The pivot table has an option to show hyperlink option for all cells that are currently in display. To do so, user need to set [ShowHyperlink](#) to **true**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .HyperlinkSettings(hyperlinksettings =>
hyperlinksettings.ShowHyperlink(true)).Render()
```

### ALLCELLS.CS

```
public ActionResult Index()
{
var data = GetPivotData();
```

```

 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Hyperlink for row headers

The pivot table has an option to show hyperlink option for row header cells alone that are currently in display. To do so, user need to set [ShowRowHeaderHyperlink](#) to **true**.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).HyperlinkSettings(hyperlinksettings =>
hyperlinksettings.ShowRowHeaderHyperlink(true)).Render()

```

### ROWHEADER.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Hyperlink for column headers

The pivot table has an option to show hyperlink option for column header cells alone that are currently in display. To do so, user need to set [ShowColumnHeaderHyperlink](#) to **true**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).HyperlinkSettings(hyperlinksettings =>
hyperlinksettings.ShowColumnHeaderHyperlink(true)).Render()
```

### COLUMNHEADER.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

### Hyperlink for value cells

The pivot table has an option to show hyperlink option for value cells alone that are currently in display. To do so, user need to set [ShowValueCellHyperlink](#) to **true**.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.DrilledMembers(drilledmembers =>
{
 drilledmembers.Name("Country").Items(ViewBag.countryMembers).Add();
 drilledmembers.Name("Year").Items(ViewBag.yearMembers).Add();
}).FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
}).HyperlinkSettings(hyperlinksettings =>
hyperlinksettings.ShowValueCellHyperlink(true)).Render()
```

### VALUECELLS.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.yearMembers = new string[] { "FY 2015" };
 ViewBag.countryMembers = new string[] { "France" };
 return View();
}
```

|                | ▼ FY 2015          |                           |                    |                          |            |
|----------------|--------------------|---------------------------|--------------------|--------------------------|------------|
|                | Q1                 |                           | Q2                 |                          | Q3         |
|                | Units Sold         | Sold Amount               | Units Sold         | Sold Amount              | Units Sold |
| ▼ France       | 114                | \$177,246                 | 108                | \$172,352                |            |
| Mountain Bikes | <a href="#">31</a> | <a href="#">\$52,824</a>  | <a href="#">51</a> | <a href="#">\$86,904</a> |            |
| Road Bikes     | <a href="#">83</a> | <a href="#">\$124,422</a> | <a href="#">57</a> | <a href="#">\$85,448</a> |            |
| ▼ Germany      | 74                 | \$117,246                 | 128                | \$152,352                |            |
| Mountain Bikes | <a href="#">51</a> | <a href="#">\$92,824</a>  | <a href="#">61</a> | <a href="#">\$76,904</a> |            |

### Hyperlink for summary cells

The pivot table has an option to show hyperlink option for summary cells alone that are currently in display. To do so, user need to set [ShowSummaryCellHyperlink](#) to true.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.DrilledMembers(drilledmembers =>
{
 drilledmembers.Name("Country").Items(ViewBag.countryMembers).Add();
 drilledmembers.Name("Year").Items(ViewBag.yearMembers).Add();
}).FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
}).HyperlinkSettings(hyperlinksettings =>
hyperlinksettings.ShowSummaryCellHyperlink(true)).Render()
```

### SUMMARYCELLS.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.yearMembers = new string[] { "FY 2015" };
 ViewBag.countryMembers = new string[] { "France" };
```

```
return View();
}
```

|                | FY 2015    |             |            |             |            |
|----------------|------------|-------------|------------|-------------|------------|
|                | Q1         |             | Q2         |             | Q3         |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France         | 114        | \$177,246   | 108        | \$172,352   |            |
| Mountain Bikes | 31         | \$52,824    | 51         | \$86,904    |            |
| Road Bikes     | 83         | \$124,422   | 57         | \$85,448    |            |
| Germany        | 74         | \$117,246   | 128        | \$152,352   |            |
| Mountain Bikes | 51         | \$92,824    | 61         | \$76,904    |            |

### Condition based hyperlink

The pivot table has an option to show hyperlink in the cells based on specific conditions. It can be configured using the [ConditionalSettings](#) class through code behind, during initial rendering. The settings required are:

- [Measure](#): Specifies the value field name, in-order to set the visibility of hyperlink for the same when condition is met.
- [Conditions](#): Specifies the operator type such as [Condition.Equals](#), [Condition.GreaterThan](#), [Condition.LessThan](#), etc.
- [Value1](#): Specifies the start value.
- [Value2](#): Specifies the end value.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.DrilledMembers(drilledmembers =>
{
drilledmembers.Name("Country").Items(ViewBag.countryMembers).Add();
drilledmembers.Name("Year").Items(ViewBag.yearMembers).Add();
}).FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
```

```
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).HyperlinkSettings(hyperlinksettings => hyperlinksettings
 .ConditionalSettings(format =>
 {
 format.Conditions(Syncfusion.EJ2.PivotView.Condition.Between).Measure("Sold")
).Value1(100).Value2(200).Add();
 })).Render();
}
```

## CONDITIONS.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.yearMembers = new string[] { "FY 2015" };
 ViewBag.countryMembers = new string[] { "France" };
 return View();
}
```

|                | FY 2015    |             |            |             |            |
|----------------|------------|-------------|------------|-------------|------------|
|                | Q1         |             | Q2         |             | Q3         |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France         | 114        | \$177,246   | 108        | \$172,352   |            |
| Mountain Bikes | 31         | \$52,824    | 51         | \$86,904    |            |
| Road Bikes     | 83         | \$124,422   | 57         | \$85,448    |            |
| Germany        | 74         | \$117,246   | 128        | \$152,352   |            |
| Mountain Bikes | 51         | \$92,824    | 61         | \$76,904    |            |

### Condition based hyperlink for specific row or column

You can apply conditions for specific row or column using [Label](#) option to show hyperlink option in the pivot table. It can be configured using the [ConditionalSettings](#) option through code behind, during initial rendering. The required settings are:

- [Label](#): Specifies the header name to get visibility of hyperlink option for row or column.
- [Conditions](#): Specifies the operator type such as [Condition.Equals](#), [Condition.GreaterThan](#), [Condition.LessThan](#), etc.
- [Value1](#): Specifies the start value.
- [Value2](#): Specifies the end value.

## CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
```

```

dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .DrilledMembers(drilledmembers =>
 {
 drilledmembers.Name("Country").Items(ViewBag.countryMembers).Add();
 drilledmembers.Name("Year").Items(ViewBag.yearMembers).Add();
 }).FormatSettings(formatsettings =>
 {

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).HyperlinkSettings(hyperlinksettings => hyperlinksettings
 .ConditionalSettings(format =>
 {

format.Conditions(Syncfusion.EJ2.PivotView.Condition.LessThan).Label("France")
 .Value1(1000).Add();
 })).Render()

```

### LABELCONDITIONS.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.yearMembers = new string[] { "FY 2015" };
 ViewBag.countryMembers = new string[] { "France" };
 return View();
}

```

### Header based hyperlink

The pivot table has an option to show hyperlink in the cells based on specific row or column header. It can be configured using the [HeaderText](#) option through code behind, during initial rendering.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .DrilledMembers(drilledmembers =>
 {
 drilledmembers.Name("Country").Items(ViewBag.countryMembers).Add();
 drilledmembers.Name("Year").Items(ViewBag.yearMembers).Add();
 }

```

```

 }).FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).HyperlinkSettings(hyperlinksettings => hyperlinksettings.HeaderText("FY 2015.Q1.Units Sold")).Render()

```

## HEADERS.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 ViewBag.yearMembers = new string[] { "FY 2015" };
 ViewBag.countryMembers = new string[] { "France" };
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |
|---------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1          |

## Event

The event [HyperlinkCellClick](#) fires on every hyperlink cell click.

It has following parameters - `cancel` and `currentCell`. The parameter `currentCell` is used to customize the host cell element by any means. Meanwhile, when the parameter `cancel` is set to `true`, applied customization will not be updated to the host cell element.

## CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)

```

```

.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).HyperLinkCellClick("hyperlink").HyperlinkSettings(hyperlinksettings =>
hyperlinksettings.ShowRowHeaderHyperlink(true)).Render()
<script>
function hyperlink(args) {
 args.cancel = false;
 args.currentCell.setAttribute("data-url",
"https://ej2.syncfusion.com/");//here we have redirected to EJ2 Syncfusion
on hyperlinkcell click
}
</script>

```

### EVENT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Toolbar in ASP.NET MVC Syncfusion Pivot Table Control

Toolbar option allows to access the frequently used features like switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc... with ease at runtime. This option can be enabled by setting the [ShowToolbar](#) property in [PivotView](#) class to **true**. The [Toolbar](#) property in [PivotView](#) class accepts the collection of built-in toolbar options.

#### Built-in Toolbar Options

The following table shows built-in toolbar options and its actions.

| Built-in Toolbar Options | Actions                    |
|--------------------------|----------------------------|
| ----- -----              |                            |
| New                      | Creates a new report       |
| Save                     | Saves the current report   |
| Save As                  | Save as current report     |
| Rename                   | Renames the current report |



- | Delete | Deletes the current report |
- | Load | Loads any report from the report list |
- | Grid | Shows pivot table |
- | Chart | Shows a chart in any type from the built-in list and option to enable/disable multiple axes |
- | Exporting | Exports the pivot table as PDF/Excel/CSV and the pivot chart as PDF and image |
- | Sub-total | Shows or hides sub totals |
- | Grand Total | Shows or hides grand totals |
- | Conditional Formatting | Shows the conditional formatting pop-up to apply formatting |
- | Number Formatting | Shows the number formatting pop-up to apply number formatting |
- | Field List | Shows the fieldlist pop-up |
- | MDX | Shows the MDX query that was run to retrieve data from the OLAP data source. **This applies only to the OLAP data source.** |

**Note:** The order of toolbar options can be changed by simply moving the position of items in the **ToolbarItems** collection. Also if end user wants to remove any toolbar option from getting displayed, it can be simply ignored from adding into the **ToolbarItems** collection.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
"NumberFormatting", "FieldList"
}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport")
```

```

t").RenameReport("renameReport").RemoveReport("removeReport").NewReport("new
Report").Render()
<style>
 #pivotview {
 width: 100%;
 height: 100%;
 }
 .e-tool-expand::before {
 content: '\e702';
 }
</style>
<script>
 function saveReport(args) {
 var reports = [];
 var isSaved = false;
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reports = JSON.parse(localStorage.pivotviewReports);
 }
 if (args.report && args.reportName && args.reportName != '') {
 reports.map(function (item) {
 if (args.reportName === item.reportName) {
 item.report = args.report;
 isSaved = true;
 }
 });
 if (!isSaved) {
 reports.push(args);
 }
 localStorage.pivotviewReports = JSON.stringify(reports);
 }
 }
 function fetchReport(args) {
 var reportCollection = [];
 var reeportList = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 reeportList.push(item.reportName);
 });
 args.reportName = reeportList;
 }
 function loadReport(args) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 args.report = item.report;
 }
 });
 }

```

```

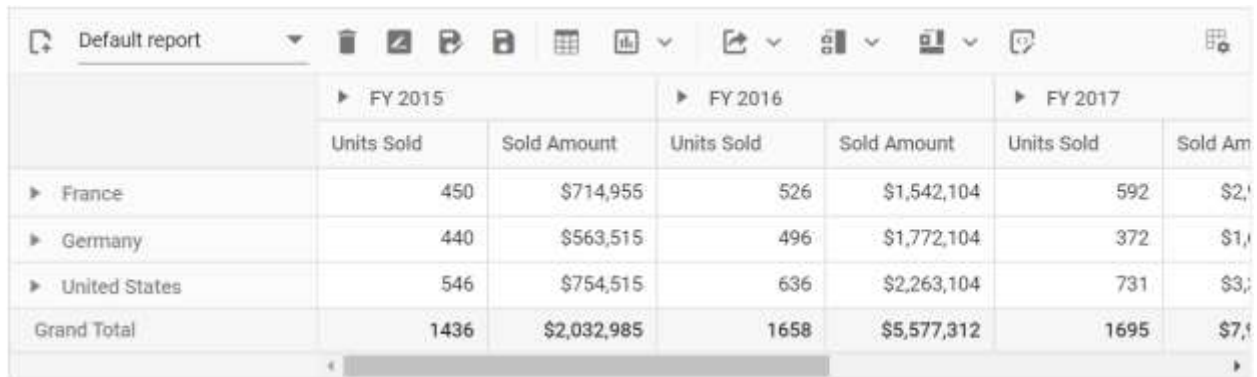
 if (args.report) {
 pivotObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
 }
 }
 function removeReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 for (var i = 0; i < reportCollection.length; i++) {
 if (reportCollection[i].reportName === args.reportName) {
 reportCollection.splice(i, 1);
 }
 }
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
 }
 function renameReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 item.reportName = args.rename;
 }
 });
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
 }
 function newReport() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.setProperties({
 dataSourceSettings: {
 columns: [],
 rows: [],
 values: [],
 filters: []
 }
 }, false);
 }
}
</script>

```

**TOOLBAR.CS**

```
public ActionResult Index()
```

```
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



|               | FY 2015    |             | FY 2016    |             | FY 2017    |             |
|---------------|------------|-------------|------------|-------------|------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 592        | \$2,100,000 |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 372        | \$1,100,000 |
| United States | 546        | \$754,515   | 636        | \$2,263,104 | 731        | \$3,100,000 |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1695       | \$7,100,000 |

### Show desired chart types in the dropdown menu

By default, all chart types are displayed in the dropdown menu included in the toolbar. However, based on the request for an application, we may need to show selective chart types on our own. This can be achieved using the [ChartTypes](#) property. To know more about supporting chart types, [click here](#).

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolbar(true).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
 ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
 Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>() { "Grid", "Chart" }).ChartTypes(new List<string>() { "Column", "Bar",
 "Line", "Area" }).Render()
<style>
 #pivotview {
 width: 100%;
```

```

 height: 100%;
 }
 .e-tool-expand::before {
 content: '\e702';
 }
</style>

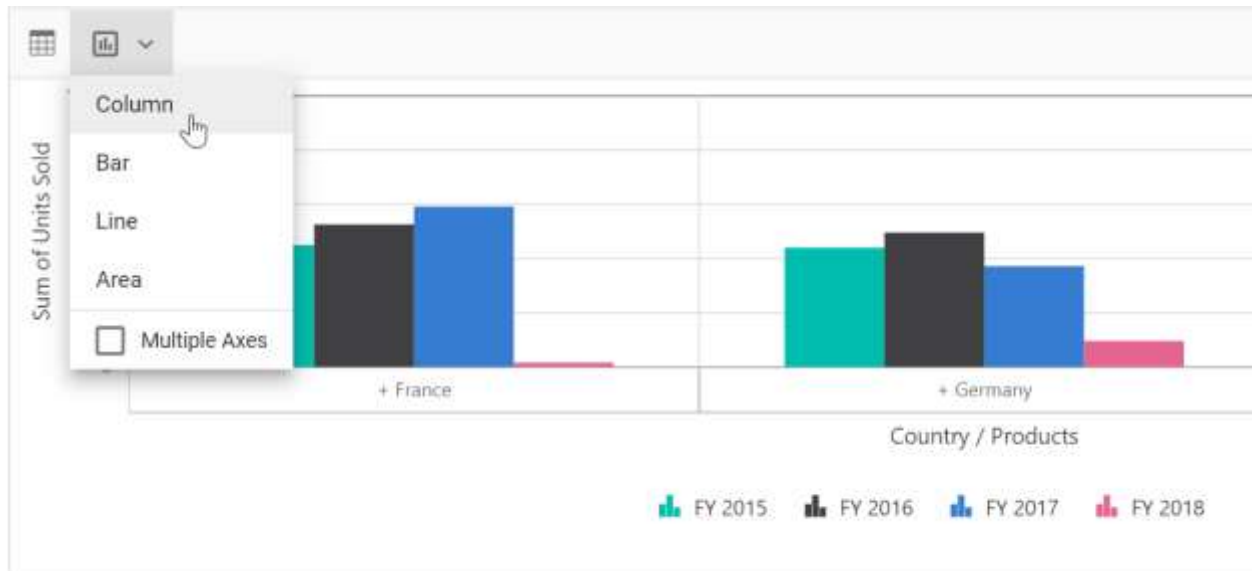
```

### TOOLBAR.CS

```

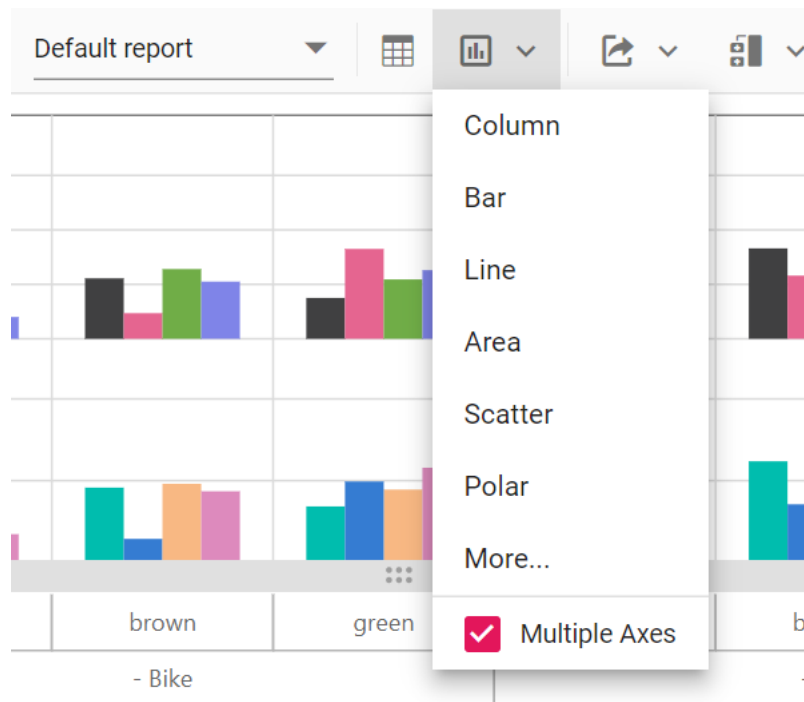
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



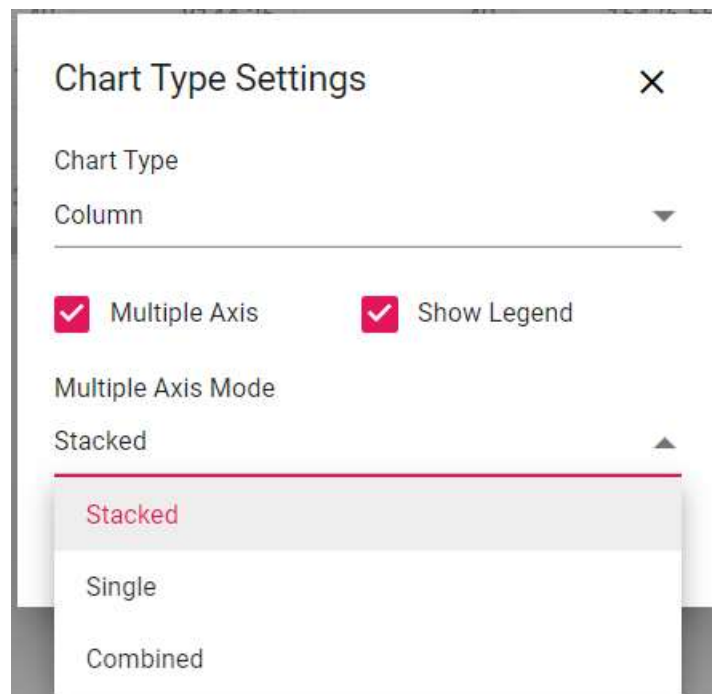
#### Switch the chart to multiple axes

In the chart, the user can switch from single axis to multiple axes with the help of the built-in checkbox available inside the chart type dropdown menu in the toolbar. For more information [refer here](#).



<!-- markdownlint-disable MD009 -->

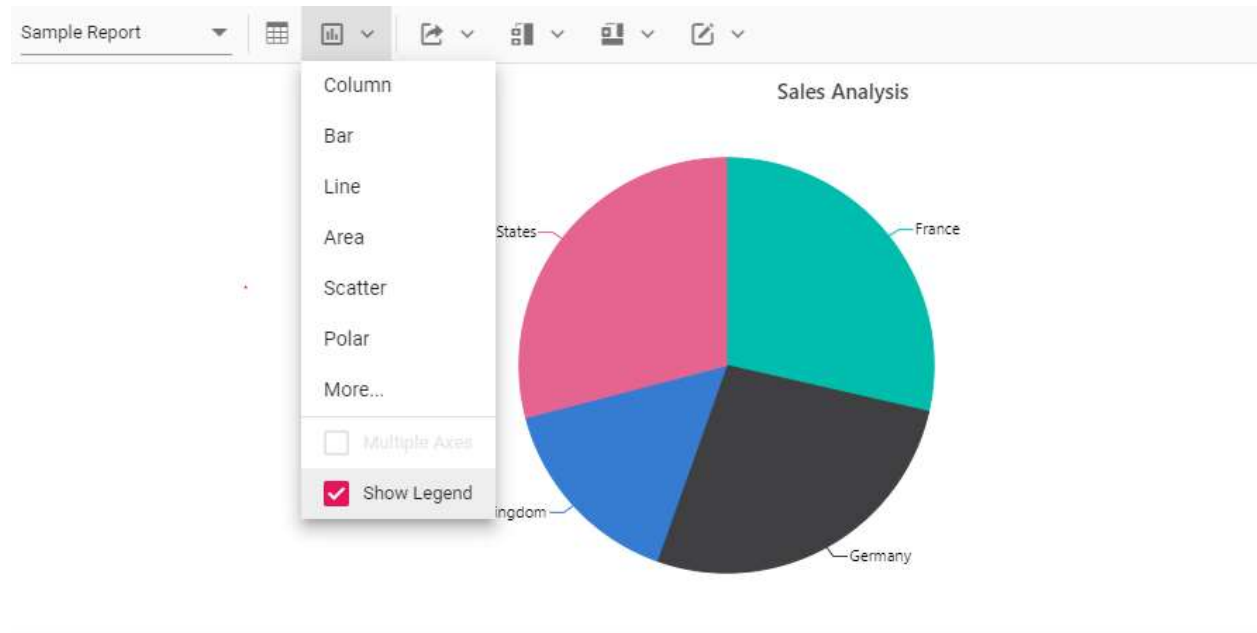
There are three modes available in **Multiple Axis** option: **Stacked**, **Single** and **Combined**. The modes can be changed using “Multiple Axis Mode” drop-down list which appears while clicking the **More...** option.



#### Show or hide legend

In the chart, legend can be shown or hidden dynamically with the help of the built-in option available in the chart type drop-down menu.

**Note:** By default, the legend is not be visible for the accumulation chart types like pie, doughnut, pyramid, and funnel. Users can enable or disable using the built-in checkbox option.



#### Adding custom option to the toolbar

In addition to the existing built-in toolbar items, new toolbar item(s) may also be included. This can be achieved by using the [ToolbarRender](#) event. The action of the new toolbar item(s) can also be defined within this event.

**Note:** The new toolbar item(s) can be added to the desired position in the toolbar using the `splice` option.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("300").ShowToolbar(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).EnableSorting(true)
.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add(); })
.Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
.Columns(columns => { columns.Name("Year").Add();
columns.Name("Order_Source").Caption("Order Source").Add(); })
.Values(values =>
{
values.Name("In_Stock").Caption("In Stock").Add();
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
.Filters(filters =>
{
filters.Name("Product_Categories").Caption("Product Categories").Add();
```

```

 })).GridSettings(new PivotViewGridSettings { ColumnWidth = 140
 }).DisplayOption(new PivotViewDisplayOption { View = View.Both
 }).Toolbar(new List<string>
 () { "Expand/Collapse" }).ToolbarRender("beforeToolbarRender").Render()
<style>
 #pivotview {
 width: 100%;
 height: 100%;
 }
 .e-tool-expand::before {
 content: '\e702';
 }
</style>
<script>
 function beforeToolbarRender(args) {
 args.customToolbar.splice(12, 0, {
 prefixIcon: 'e-tool-expand e-icons', tooltipText:
'Expand/Collapse',
 click: function (args) {
 var pivotTableObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotTableObj.dataSourceSettings.expandAll =
!pivotTableObj.dataSourceSettings.expandAll;
 },
 });
 }
</script>

```

### TOOLBARCUSTOMIZE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |
|---------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 5          |
| Germany       | 440        | \$563,515   | 4          |
| United States | 546        | \$754,515   | 6          |
| Grand Total   | 1436       | \$2,032,985 | 16         |



In the above topic, we have seen how to add an icon as one of the toolbar item in toolbar panel. In the next topic, we are going to see how to frame the entire toolbar panel and how to add a custom control in it.

### Toolbar Template

It allows to customize the toolbar panel by using template option. It allows any custom control to be used as one of the toolbar item inside the toolbar panel. It can be achieved by two ways.

Here, the entire toolbar panel can be framed in HTML elements that are appended at the top of the pivot table. The **id** of the HTML element needs to be set in the [toolbarTemplate](#) property in-order to map it to the pivot table.

### CSHTML

```
<div id="template">
 <div>
 @Html.EJS().Button("expandbtn").Content("Expand
All").CssClass("e-flat").IsPrimary(true).Render()
 @Html.EJS().Button("collapsebtn").Content("Collapse
ALL").CssClass("e-flat").IsPrimary(true).Render()
 </div>
</div>

@Html.EJS().PivotView("PivotGrid").Width("100%").Height("300").ShowToolbar(t
rue).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>
) ViewBag.Data)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Columns(columns =>
{
columns.Name("Year").Add(); columns.Name("Quarter").Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(new
Syncfusion.EJ2.PivotView.PivotViewGridSettings { ColumnWidth = 140
}).ToolbarTemplate("#template").Render()
<script>
 document.getElementById("expandbtn").addEventListener('click', function
() {
 var pivotObj =
document.getElementById("PivotGrid").ej2_instances[0];
 pivotObj.dataSourceSettings.expandAll = true;
 });
 document.getElementById("collapsebtn").addEventListener('click',
function () {
 var pivotObj =
document.getElementById("PivotGrid").ej2_instances[0];
 pivotObj.dataSourceSettings.expandAll = false;
```

```
});
</script>
```

### TOOLBARCUSTOMIZE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

| EXPAND ALL    COLLAPSE ALL |            |               |            |               |            |             |
|----------------------------|------------|---------------|------------|---------------|------------|-------------|
|                            | FY 2015    |               | FY 2016    |               | FY 2017    |             |
|                            | Units Sold | Sold Amount   | Units Sold | Sold Amount   | Units Sold | Sold Amount |
| France                     | 729        | \$1,160,099.5 | 609        | \$983,317     | 703        | \$1,14      |
| Germany                    | 528        | \$845,472     | 667        | \$1,067,220   | 579        | \$94        |
| United Kingdom             | 782        | \$1,263,109.5 | 640        | \$1,031,630.5 | 657        | \$1,041     |
| United States              | 682        | \$1,085,398.5 | 480        | \$770,362     | 644        | \$1,022     |
| Grand Total                | 2721       | \$4,354,079.5 | 2396       | \$3,852,529.5 | 2583       | \$4,11      |

Another option allows to frame a custom toolbar item using HTML elements and include in the toolbar panel at the desired position. The custom toolbar items can be declared as control **instance** or element **id** in the [toolbar](#) property in pivot table.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@{
 List<object> toolbarItems = new List<object>();
 toolbarItems.Add(new { template = "#enablertl" });
 toolbarItems.Add(new { template = "#disablertl" });
}
@Html.EJS().Button("enablertl").Content("Enable Rtl").CssClass("e-flat").IsPrimary(true).Render()
@Html.EJS().Button("disablertl").Content("Disable Rtl").CssClass("e-flat").IsPrimary(true).Render()

@Html.EJS().PivotView("PivotGrid").Width("100%).Height("300").ShowToolbar(true).ShowToolbar(true).DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>
)ViewBag.Data)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
 }).Rows(rows =>
 {
```

```

 rows.Name("Country").Caption("Products").Add();
rows.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
 })).GridSettings(new
Syncfusion.EJ2.PivotView.PivotViewGridSettings { ColumnWidth = 140
}).Toolbar(toolbarItems).Render()
</div>
</div>
<script>
 document.getElementById("enablertl").addEventListener('click', function ()
 {
 var pivotObj =
document.getElementById("PivotGrid").ej2_instances[0];
 pivotObj.enableRtl = true;
 });
 document.getElementById("disablertl").addEventListener('click', function
() {
 var pivotObj =
document.getElementById("PivotGrid").ej2_instances[0];
 pivotObj.enableRtl = false;
 });
</script>

```

### TOOLBARCUSTOMIZE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

| ENABLE RTL    DISABLE RTL |            |               |            |               |            |             |
|---------------------------|------------|---------------|------------|---------------|------------|-------------|
|                           | FY 2015    |               | FY 2016    |               | FY 2017    |             |
|                           | Units Sold | Sold Amount   | Units Sold | Sold Amount   | Units Sold | Sold Amount |
| France                    | 729        | \$1,160,999.5 | 609        | \$983,317     | 703        | \$1,14      |
| Germany                   | 528        | \$845,472     | 667        | \$1,067,220   | 579        | \$94        |
| United Kingdom            | 782        | \$1,263,109.5 | 640        | \$1,031,630.5 | 657        | \$1,041     |
| United States             | 682        | \$1,085,398.5 | 480        | \$770,362     | 644        | \$1,022     |
| Grand Total               | 2721       | \$4,354,079.5 | 2396       | \$3,852,529.5 | 2583       | \$4,11      |

**Note:** For both options, the actions for the toolbar template items can be defined in the event [toolbarClick](#). Also, if the toolbar item is a custom control then its built-in events can also be accessed.

<!-- markdownlint-disable MD009 -->

### Save and load report as a JSON file

The current pivot report can be saved as a JSON file in the desired path and loaded back to the pivot table at any time.

**CSHTML**

```

@using Syncfusion.EJ2.PivotView
Save<div class="fileUpload btn btn-
primary">Load<input id="files" type="file" class="upload"
/></div>
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").DataSourceSet
tings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).EnableSorting
(true)
.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add(); })
.Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
.Columns(columns => { columns.Name("Year").Add();
columns.Name("Order_Source").Caption("Order Source").Add(); })
.Values(values =>
{
values.Name("In_Stock").Caption("In Stock").Add();
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
.Filters(filters =>
{
filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).DataBound("ondataBound").Render()
<style>
#pivotview {
width: 100%;
height: 100%;
}
.fileUpload {
position: relative;
overflow: hidden;
margin: 10px;
}
.fileUpload input.upload {
position: absolute;
top: 0;
right: 0;
margin: 0;
padding: 0;
font-size: 20px;
cursor: pointer;
opacity: 0;
filter: alpha(opacity=0);
}
</style>
<script>
function ondataBound(args) {
var pivotTableObj =
document.getElementById('pivotview').ej2_instances[0];
var dataSource =
JSON.parse(pivotTableObj.getPersistData()).dataSourceSettings.dataSource;
var a = document.getElementById('save');

```

```

 var mime_type = 'application/octet-stream'; // text/html, image/png,
et c
 a.setAttribute('download', 'pivot.JSON');
 a.href = 'data:' + mime_type + ';base64,' +
 btoa(JSON.stringify(dataSource) || '');
 document.getElementById('files').addEventListener('change',
 readBlob, false);
 }
 function readBlob(args) {
 var files = document.getElementById('load').files;
 var file = files[0];
 var start = 0;
 var stop = file.size - 1;
 var reader = new FileReader();
 reader.onloadend = function(evt) {
 if (evt.target.readyState == FileReader.DONE) {
 var pivotTableObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotTableObj.dataSource = JSON.parse(evt.target.result);
 }
 };
 var blob = file.slice(start, stop + 1);
 reader.readAsBinaryString(blob);
 }
}
</script>

```

### SAVE-LOAD-JSON.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Save and load reports to a SQL database

SQL Server is a relational database management system (RDBMS) that can be used to store and manage large amounts of data. In this topic, we will see how to save, save as, rename, load, delete, and add reports between a SQL Server database and an ASP.NET MVC Pivot Table at runtime.

#### Create a Web API service to connect to a SQL Server database

1. Open Visual Studio and create an ASP.NET Core Web App project type, naming it **MyWebService**. To create an ASP.NET Core Web application, follow the document [link](#).

Configure your new project

ASP.NET Core Web App C# Linux macOS Windows Cloud Service Web

Project name

MyWebService

Location

C:\Users\username\source\repos

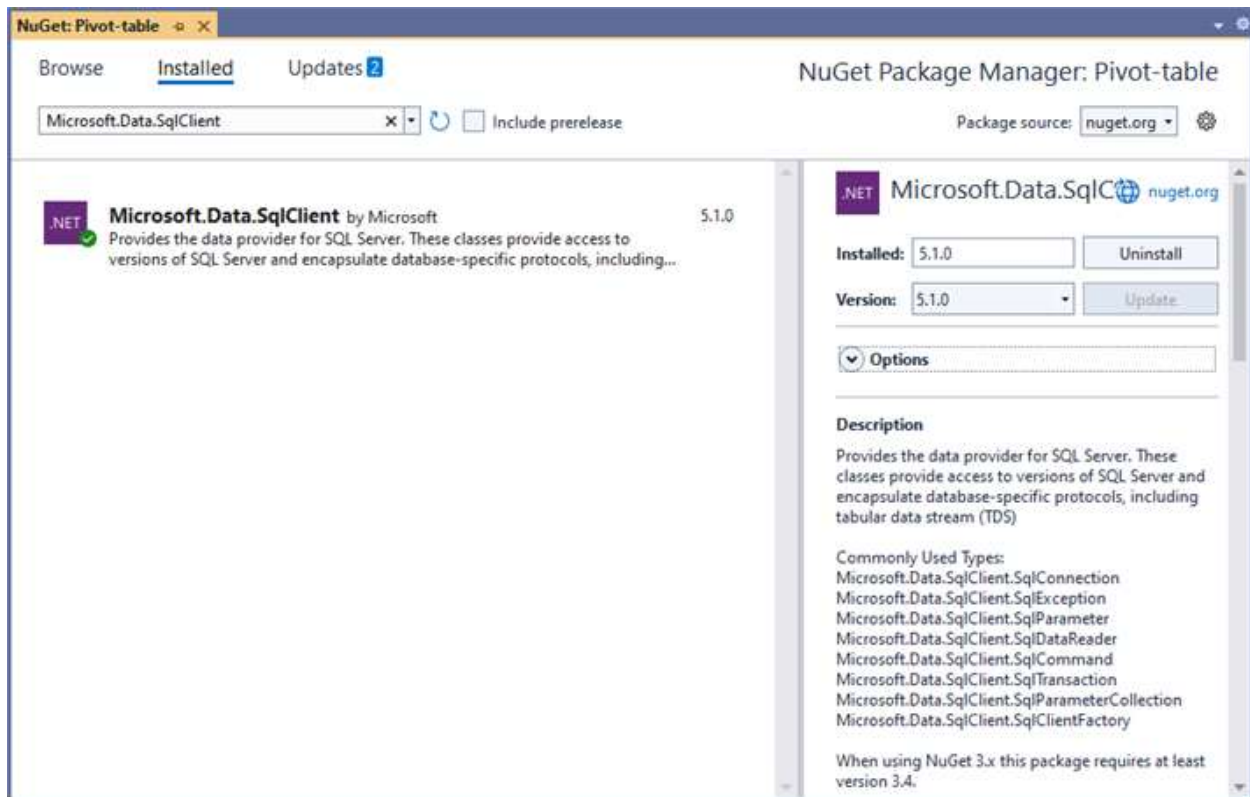
Solution name ⓘ

MyWebService

☐ Place solution and project in the same directory

Back Next

2. To connect a SQL Server database using the Microsoft SqlClient in our application, we need to install the [Microsoft.Data.SqlClient](#) NuGet package. To do so, open the NuGet package manager of the project solution, search for the package **Microsoft.Data.SqlClient** and install it.



3. Under the **Controllers** folder, create a Web API controller (aka, PivotController.cs) file that aids in data communication with the Pivot Table.

4. In the Web API Controller (aka, PivotController), the **OpenConnection** method is used to connect to the SQL database. The **GetDataTable** method then processes the specified SQL query string, retrieves data from the database, and converts it into a **DataTable** using **SqlCommand** and **SqlDataAdapter**. This **DataTable** can be used to retrieve saved reports and modify them further as shown in the code block below.

[PivotController.cs]

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebService.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpPost]
```

```
[Route("Pivot/SaveReport")]
public void SaveReport([FromBody] Dictionary<string, string> reportArgs)
{
 SaveReportToDB(reportArgs["reportName"], reportArgs["report"]);
}

[HttpPost]
[Route("Pivot/FetchReport")]
public IActionResult FetchReport()
{
 return Ok((FetchReportListFromDB()));
}

[HttpPost]
[Route("Pivot/LoadReport")]
public IActionResult LoadReport([FromBody] Dictionary<string, string> reportArgs)
{
 return Ok((LoadReportFromDB(reportArgs["reportName"])));
}

[HttpPost]
[Route("Pivot/RemoveReport")]
public void RemoveReport([FromBody] Dictionary<string, string> reportArgs)
{
 RemoveReportFromDB(reportArgs["reportName"]);
}

[HttpPost]
[Route("Pivot/RenameReport")]
public void RenameReport([FromBody] RenameReportDB reportArgs)
{
 RenameReportInDB(reportArgs.ReportName, reportArgs.RenameReport, reportArgs.isReportExists);
}

public class RenameReportDB
{
 public string ReportName { get; set; }
 public string RenameReport { get; set; }
}
```



```
public bool isReportExists { get; set; }
}

private void SaveReportToDB(string reportName, string report)
{
 SqlConnection sqlConn = OpenConnection();
 bool isDuplicate = true;
 SqlCommand cmd1 = null;
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 isDuplicate = false;
 cmd1 = new SqlCommand("update ReportTable set Report=@Report where ReportName like @ReportName", sqlConn);
 }
 }
 if (isDuplicate)
 {
 cmd1 = new SqlCommand("insert into ReportTable (ReportName,Report) Values(@ReportName,@Report)", sqlConn);
 }
 cmd1.Parameters.AddWithValue("@ReportName", reportName);
 cmd1.Parameters.AddWithValue("@Report", report.ToString());
 cmd1.ExecuteNonQuery();
 sqlConn.Close();
}

private string LoadReportFromDB(string reportName)
{
 SqlConnection sqlConn = OpenConnection();
 string report = string.Empty;
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
```

```
report = (string)row["Report"];
break;
}
}
sqlConn.Close();
return report;
}
private List<string> FetchReportListFromDB()
{
 SqlConnection sqlConn = OpenConnection();
 List<string> reportNames = new List<string>();
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if (!string.IsNullOrEmpty(row["ReportName"] as string))
 {
 reportNames.Add(row["ReportName"].ToString());
 }
 }
 sqlConn.Close();
 return reportNames;
}
private void RenameReportInDB(string reportName, string renameReport, bool isReportExists)
{
 SqlConnection sqlConn = OpenConnection();
 SqlCommand cmd1 = null;
 if (isReportExists)
 {
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
 sqlConn);
 }
 }
 }
}
```

```
break;
}
}
cmd1.ExecuteNonQuery();
}
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
 if ((row["ReportName"] as string).Equals(reportName))
 {
 cmd1 = new SqlCommand("update ReportTable set ReportName=@RenameReport where ReportName
like '%" + reportName + "%'", sqlConn);
 break;
 }
}
cmd1.Parameters.AddWithValue("@RenameReport", renameReport);
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private void RemoveReportFromDB(string reportName)
{
 SqlConnection sqlConn = OpenConnection();
 SqlCommand cmd1 = null;
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
sqlConn);
 break;
 }
 }
 cmd1.ExecuteNonQuery();
 sqlConn.Close();
}
```

```

private SqlConnection OpenConnection()
{
 // Replace with your own connection string.
 string connectionString = @"<Enter your valid connection string here>";
 SqlConnection sqlConn = new SqlConnection(connectionString);
 sqlConn.Open();
 return sqlConn;
}

private DataTable GetDataTable(SqlConnection sqlConn)
{
 string xquery = "select * from ReportTable";
 SqlCommand cmd = new SqlCommand(xquery, sqlConn);
 SqlDataAdapter da = new SqlDataAdapter(cmd);
 DataTable dt = new DataTable();
 da.Fill(dt);
 return dt;
}
}

```

5. When you run the app, it will be hosted at <https://localhost:44313>. You can use the hosted URL to save and load reports in the SQL database from the Pivot Table.

Further, let us explore more on how to save, load, rename, delete, and add reports using the built-in toolbar options via Web API controller (aka, PivotController) one-by-one.

#### [Saving a report](#)

When you select the **“Save a report”** option from the toolbar, the [saveReport](#) event is triggered. In this event, an AJAX request is made to the Web API controller's **SaveReport** method, passing the name of the current report and the current report, which you can use to check and save in the SQL database.

For example, the report shown in the following code snippet will be passed to the **SaveReport** method along with the report name **“Sample Report”** and saved in the SQL database.

[Index.cshtml]

```
`csharp
```

```
@using Syncfusion.EJ2.PivotView;
```

```
@Html.EJS().PivotView("pivotview").Width("700").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowConditionalFormatting(true).AllowPdfExport(true).AllowCalculatedField(true).DataSource
```

```

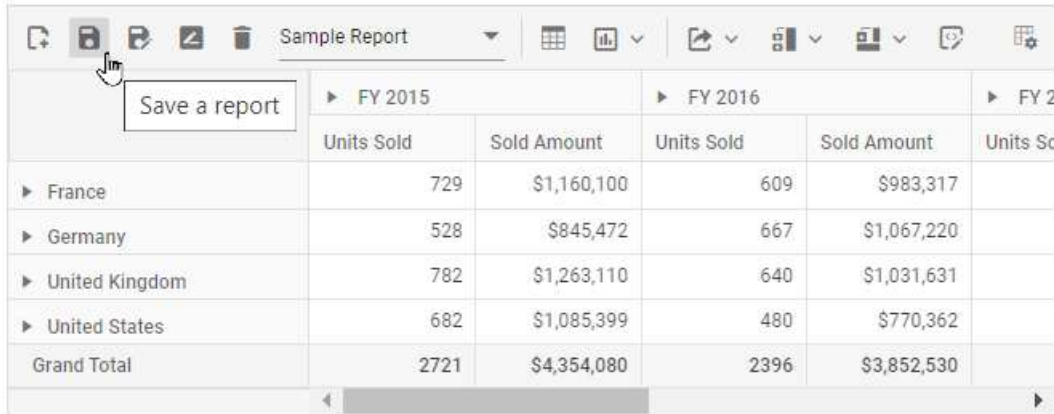
Settings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
.FormatSettings(formatSettings => { formatSettings.Name("Amount").Format("C0").Add(); })
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Production Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add(); values.Name("Amount").Caption("Sold
Amount").Add();
}).DisplayOption(new PivotViewDisplayOption { View = View.Both }).ChartSettings(new
PivotViewChartSettings { Value = "Amount", EnableExport = true, EnableMultipleAxis = false
}).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load",
"Grid", "Chart", "Export", "SubTotal", "GrandTotal", "Formatting",
"FieldList"}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameR
eport("renameReport").RemoveReport("removeReport").NewReport("newReport").ToolbarRender("bef
oreToolbarRender").Render()
<script>
function saveReport(args) {
var report = JSON.parse(args.report);
report.dataSourceSettings.dataSource = [];
fetch('https://localhost:44313/Pivot/SaveReport', {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName, report: JSON.stringify(report) })
}).then(response => {
fetchReport(args);
});

```

```
}
</script>
`

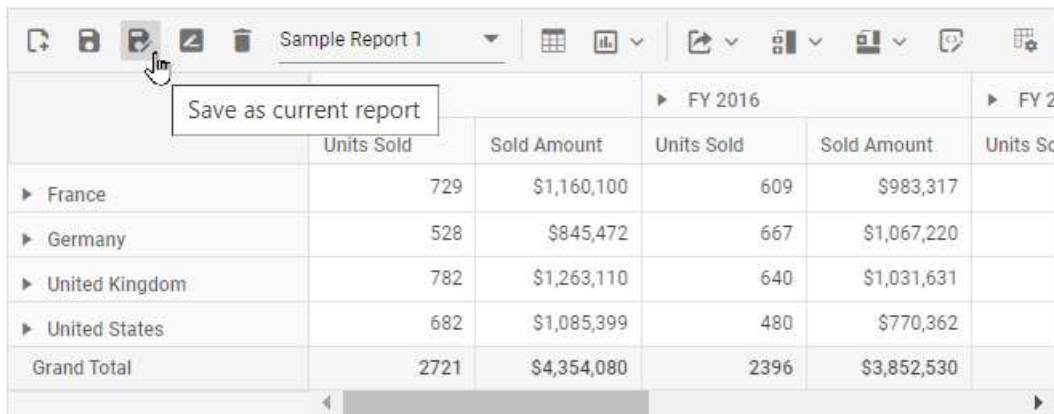
[PivotController.cs]
`csharp
namespace MyWebApp.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpPost]
 [Route("Pivot/SaveReport")]
 public void SaveReport([FromBody] Dictionary<string, string> reportArgs)
 {
 SaveReportToDB(reportArgs["reportName"], reportArgs["report"]);
 }
 private void SaveReportToDB(string reportName, string report)
 {
 SqlConnection sqlConn = OpenConnection();
 bool isDuplicate = true;
 SqlCommand cmd1 = null;
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 isDuplicate = false;
 cmd1 = new SqlCommand("update ReportTable set Report=@Report where ReportName like @ReportName", sqlConn);
 }
 }
 if (isDuplicate)
 {
```

```
cmd1 = new SqlCommand("insert into ReportTable (ReportName,Report)
Values(@ReportName,@Report)", sqlConn);
}
cmd1.Parameters.AddWithValue("@ReportName", reportName);
cmd1.Parameters.AddWithValue("@Report", report.ToString());
cmd1.ExecuteNonQuery();
sqlConn.Close();
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}
```



|                | FY 2015    |             | FY 2016    |             | FY 2017    |
|----------------|------------|-------------|------------|-------------|------------|
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France         | 729        | \$1,160,100 | 609        | \$983,317   |            |
| Germany        | 528        | \$845,472   | 667        | \$1,067,220 |            |
| United Kingdom | 782        | \$1,263,110 | 640        | \$1,031,631 |            |
| United States  | 682        | \$1,085,399 | 480        | \$770,362   |            |
| Grand Total    | 2721       | \$4,354,080 | 2396       | \$3,852,530 |            |

In the meantime, you can save a duplicate of the current report to the SQL Server database with a different name by selecting **“Save as current report”** from the toolbar. The [saveReport](#) event will then be triggered with the new report name **“Sample Report 1”** and the current report. You can save them to the SQL Server database after passing them to the Web API service, as mentioned above.



|                | FY 2015    |             | FY 2016    |             | FY 2017    |
|----------------|------------|-------------|------------|-------------|------------|
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France         | 729        | \$1,160,100 | 609        | \$983,317   |            |
| Germany        | 528        | \$845,472   | 667        | \$1,067,220 |            |
| United Kingdom | 782        | \$1,263,110 | 640        | \$1,031,631 |            |
| United States  | 682        | \$1,085,399 | 480        | \$770,362   |            |
| Grand Total    | 2721       | \$4,354,080 | 2396       | \$3,852,530 |            |

#### Loading a report

When you select the dropdown menu item from the toolbar, the [loadReport](#) event is triggered. In this event, an AJAX request is made to the **LoadReport** method of the Web API controller, passing the name of the selected report. The method uses this information to search for the report in the SQL database, fetch it, and load it into the pivot table.

For example, if the report name **“Sample Report 1”** is selected from a dropdown menu and passed, the **LoadReport** method will use that name to search for the report in the SQL database, retrieve it, and then load it into the pivot table.

[Index.cshtml]

```
`csharp
```

```
@using Syncfusion.EJ2.PivotView;
```

```
@Html.EJS().PivotView("pivotview").Width("700").ShowToolBar(true).ShowFieldList(true).AllowExcelExport(true).AllowConditionalFormatting(true).AllowPdfExport(true).AllowCalculatedField(true).DataSourceSettings(dataSourceSettings =>
```

```
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
```

```
.FormatSettings(formatsettings => { formatsettings.Name("Amount").Format("C0").Add(); })
```



```

.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Production Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add(); values.Name("Amount").Caption("Sold
Amount").Add();
}).DisplayOption(new PivotViewDisplayOption { View = View.Both }).ChartSettings(new
PivotViewChartSettings { Value = "Amount", EnableExport = true, EnableMultipleAxis = false
}).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load",
"Grid", "Chart", "Export", "SubTotal", "GrandTotal", "Formatting",
"FieldList"}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameR
eport("renameReport").RemoveReport("removeReport").NewReport("newReport").ToolbarRender("bef
oreToolbarRender").Render()
<script>
function loadReport(args) {
fetch('https://localhost:44313/Pivot/LoadReport', {
method: 'POST',
headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName })
}).then(res => res.json())
.then(response => {
if (response) {
var report = JSON.parse(response);
var pivotTableObj = document.getElementById('pivotview').ej2_instances[0];
report.dataSourceSettings.dataSource = pivotTableObj.dataSourceSettings.dataSource;
pivotTableObj.dataSourceSettings = report.dataSourceSettings;
}
}

```

```
});
}
</script>
,
```

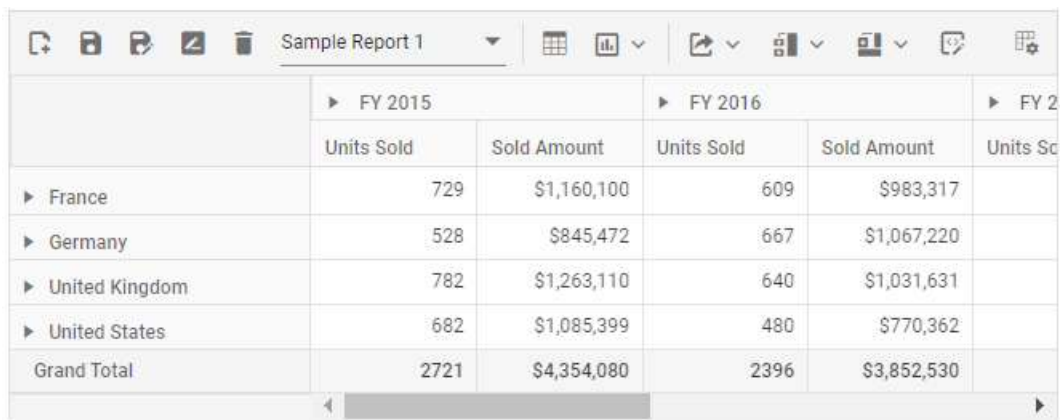
```
[PivotController.cs]
```

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebApp.Controllers
{
 [ApiController]
 [Route("[controller]")]
 public class PivotController : ControllerBase
 {
 [HttpPost]
 [Route("Pivot/LoadReport")]
 public IActionResult LoadReport([FromBody] Dictionary<string, string> reportArgs)
 {
 return Ok((LoadReportFromDB(reportArgs["reportName"])));
 }
 private string LoadReportFromDB(string reportName)
 {
 SqlConnection sqlConn = OpenConnection();
 string report = string.Empty;
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 report = (string)row["Report"];
 break;
 }
 }
 }
 }
}
```

```

sqlConn.Close();
return report;
}
private SqlConnection OpenConnection()
{
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}
private DataTable GetDataTable(SqlConnection sqlConn)
{
string xquery = "select * from ReportTable";
SqlCommand cmd = new SqlCommand(xquery, sqlConn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataTable dt = new DataTable();
da.Fill(dt);
return dt;
}
}
}

```



The screenshot shows the Syncfusion Pivot Table Control interface. At the top is a toolbar with various icons for actions like refresh, save, print, zoom, and pivot table manipulation. Below the toolbar is a table titled 'Sample Report 1'. The table has a hierarchical structure with expandable rows and columns. The columns are organized into three main sections: FY 2015, FY 2016, and FY 2017. Each section contains 'Units Sold' and 'Sold Amount' columns. The rows include 'France', 'Germany', 'United Kingdom', 'United States', and a 'Grand Total' row. The data is displayed in a grid with alternating row colors for readability.

|                | FY 2015    |             | FY 2016    |             | FY 2017    |
|----------------|------------|-------------|------------|-------------|------------|
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France         | 729        | \$1,160,100 | 609        | \$983,317   |            |
| Germany        | 528        | \$845,472   | 667        | \$1,067,220 |            |
| United Kingdom | 782        | \$1,263,110 | 640        | \$1,031,631 |            |
| United States  | 682        | \$1,085,399 | 480        | \$770,362   |            |
| Grand Total    | 2721       | \$4,354,080 | 2396       | \$3,852,530 |            |

### Renaming a report

When you select the **“Rename a current report”** option from the toolbar, the [renameReport](#) event is triggered. In this event, an AJAX request is made to the **RenameReport** method of the Web API controller, passing the current and new report names, where you can use the current report name to identify the report and resave it with the new report name in the SQL database.

For example, if we rename the current report from **“Sample Report 1”** to **“Sample Report 2”**, both **“Sample Report 1”** and **“Sample Report 2”** will be passed to the **RenameReport** method, which will rename the current report with the new report name **“Sample Report 2”** in the SQL database.

[Index.cshtml]

```
`csharp
@using Syncfusion.EJ2.PivotView;

@Html.EJS().PivotView("pivotview").Width("700").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowConditionalFormatting(true).AllowPdfExport(true).AllowCalculatedField(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
.FormatSettings(formatsettings => { formatsettings.Name("Amount").Format("C0").Add(); })
.Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Production Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add(); values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Both }).ChartSettings(new PivotViewChartSettings { Value = "Amount", EnableExport = true, EnableMultipleAxis = false
}).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load",
"Grid", "Chart", "Export", "SubTotal", "GrandTotal", "Formatting",
"FieldList"}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport").RemoveReport("removeReport").NewReport("newReport").ToolbarRender("beforeToolbarRender").Render()

<script>
function renameReport(args) {
fetch('https://localhost:44313/Pivot/RenameReport', {
method: 'POST',
```

```

headers: {
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName, renameReport: args.rename, isReportExists:
args.isReportExists })
}).then(response => {
fetchReport(args);
});
}
</script>
`

```

```
[PivotController.cs]
```

```

`csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebApp.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
[Route("Pivot/RenameReport")]
public void RenameReport([FromBody] RenameReportDB reportArgs)
{
RenameReportInDB(reportArgs.ReportName, reportArgs.RenameReport, reportArgs.isReportExists);
}
public class RenameReportDB
{
public string ReportName { get; set; }
public string RenameReport { get; set; }
}
}
}

```

```
public bool isReportExists { get; set; }
}

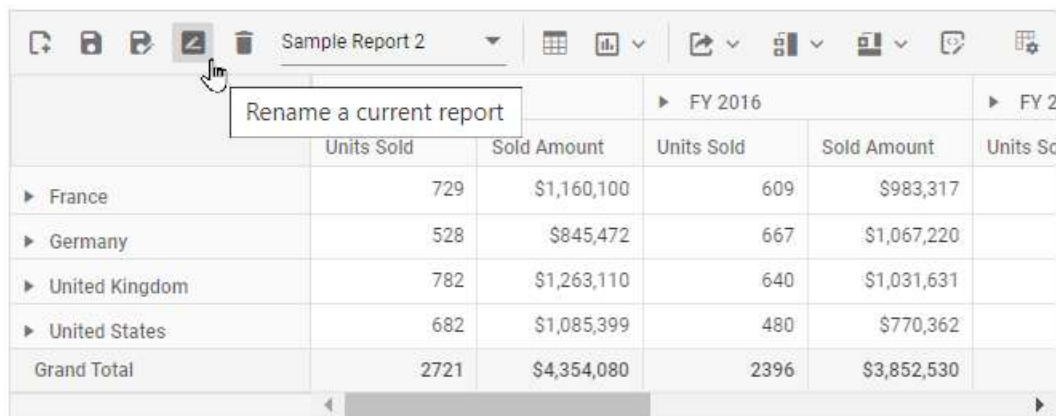
private void RenameReportInDB(string reportName, string renameReport, bool isReportExists)
{
 SqlConnection sqlConn = OpenConnection();
 SqlCommand cmd1 = null;
 if (isReportExists)
 {
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
 sqlConn);
 break;
 }
 }
 cmd1.ExecuteNonQuery();
 }
 foreach (DataRow row in GetDataTable(sqlConn).Rows)
 {
 if ((row["ReportName"] as string).Equals(reportName))
 {
 cmd1 = new SqlCommand("update ReportTable set ReportName=@RenameReport where ReportName
 like '%" + reportName + "%'", sqlConn);
 break;
 }
 }
 cmd1.Parameters.AddWithValue("@RenameReport", renameReport);
 cmd1.ExecuteNonQuery();
 sqlConn.Close();
}

private SqlConnection OpenConnection()
{

```

```
// Replace with your own connection string.
string connectionString = @"<Enter your valid connection string here>";
SqlConnection sqlConn = new SqlConnection(connectionString);
sqlConn.Open();
return sqlConn;
}

private DataTable GetDataTable(SqlConnection sqlConn)
{
 string xquery = "select * from ReportTable";
 SqlCommand cmd = new SqlCommand(xquery, sqlConn);
 SqlDataAdapter da = new SqlDataAdapter(cmd);
 DataTable dt = new DataTable();
 da.Fill(dt);
 return dt;
}
}
```



|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
|----------------|------------|-------------|------------|-------------|------------|
| France         | 729        | \$1,160,100 | 609        | \$983,317   |            |
| Germany        | 528        | \$845,472   | 667        | \$1,067,220 |            |
| United Kingdom | 782        | \$1,263,110 | 640        | \$1,031,631 |            |
| United States  | 682        | \$1,085,399 | 480        | \$770,362   |            |
| Grand Total    | 2721       | \$4,354,080 | 2396       | \$3,852,530 |            |

#### Deleting a report

When you select the **“Delete a current report”** option from the toolbar, the [removeReport](#) event is triggered. In this event, an AJAX request is made to the **RemoveReport** method of the Web API controller, passing the current report name to identify and delete the appropriate report from the SQL database.

**Note:** \* If the current report **n** from the pivot table is deleted, the pivot table will automatically load the last report from the report list.

\* When a report is removed from a pivot table with only one report, the SQL database refreshes; however, the pivot table will continue to show the removed report until a new report is added to the pivot table.

For example, if we delete the current report **“Sample Report 2”** from the pivot table, the current report name **“Sample Report 2”** is passed to the **RemoveReport** method, which allows you to identify and delete the report from the SQL database.

[Index.cshtml]

```
`csharp
```

```
@using Syncfusion.EJ2.PivotView;
```

```
@Html.EJS().PivotView("pivotview").Width("700").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowConditionalFormatting(true).AllowPdfExport(true).AllowCalculatedField(true).DataSourceSettings(dataSourceSettings =>
```

```
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.Data).ExpandAll(false)
```

```
.FormatSettings(formatsettings => { formatsettings.Name("Amount").Format("C0").Add(); })
```

```
.Rows(rows =>
```

```
{
```

```
rows.Name("Country").Add(); rows.Name("Products").Add();
```

```
}).Columns(columns =>
```

```
{
```

```
columns.Name("Year").Caption("Production Year").Add(); columns.Name("Quarter").Add();
```

```
}).Values(values =>
```

```
{
```

```
values.Name("Sold").Caption("Units Sold").Add(); values.Name("Amount").Caption("Sold Amount").Add();
```

```
}).DisplayOption(new PivotViewDisplayOption { View = View.Both }).ChartSettings(new
```

```
PivotViewChartSettings { Value = "Amount", EnableExport = true, EnableMultipleAxis = false
```

```
}).Toolbar(new List<string>
```

```
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load",
```

```
"Grid", "Chart", "Export", "SubTotal", "GrandTotal", "Formatting",
```

```
"FieldList"}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport").RemoveReport("removeReport").NewReport("newReport").ToolbarRender("beforeToolbarRender").Render()
```

```
<script>
```

```
function removeReport(args) {
```

```
fetch('https://localhost:44313/Pivot/RemoveReport', {
```

```
method: 'POST',
```

```
headers: {
```



```
'Accept': 'application/json',
'Content-Type': 'application/json',
},
body: JSON.stringify({ reportName: args.reportName })
}).then(response => {
fetchReport(args);
});
}
</script>
`
```

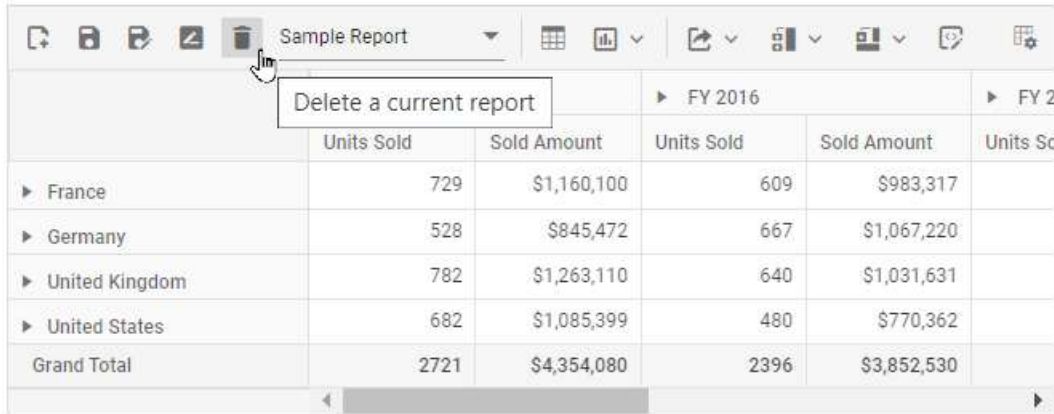
```
[PivotController.cs]
```

```
`csharp
using Microsoft.AspNetCore.Mvc;
using Microsoft.Data.SqlClient;
using System.Data;
namespace MyWebApp.Controllers
{
[ApiController]
[Route("[controller]")]
public class PivotController : ControllerBase
{
[HttpPost]
[Route("Pivot/RemoveReport")]
public void RemoveReport([FromBody] Dictionary<string, string> reportArgs)
{
RemoveReportFromDB(reportArgs["reportName"]);
}
private void RemoveReportFromDB(string reportName)
{
SqlConnection sqlConn = OpenConnection();
SqlCommand cmd1 = null;
foreach (DataRow row in GetDataTable(sqlConn).Rows)
{
```

```
if ((row["ReportName"] as string).Equals(reportName))
{
 cmd1 = new SqlCommand("delete from ReportTable where ReportName like '%" + reportName + "%'",
 sqlConn);
 break;
}
cmd1.ExecuteNonQuery();
sqlConn.Close();
}

private SqlConnection OpenConnection()
{
 // Replace with your own connection string.
 string connectionString = @"<Enter your valid connection string here>";
 SqlConnection sqlConn = new SqlConnection(connectionString);
 sqlConn.Open();
 return sqlConn;
}

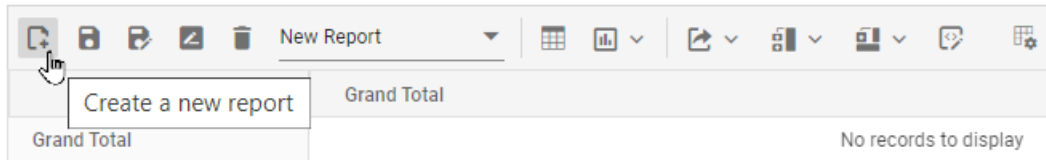
private DataTable GetDataTable(SqlConnection sqlConn)
{
 string xquery = "select * from ReportTable";
 SqlCommand cmd = new SqlCommand(xquery, sqlConn);
 SqlDataAdapter da = new SqlDataAdapter(cmd);
 DataTable dt = new DataTable();
 da.Fill(dt);
 return dt;
}
}
```



|                  | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
|------------------|------------|-------------|------------|-------------|------------|
| ► France         | 729        | \$1,160,100 | 609        | \$983,317   |            |
| ► Germany        | 528        | \$845,472   | 667        | \$1,067,220 |            |
| ► United Kingdom | 782        | \$1,263,110 | 640        | \$1,031,631 |            |
| ► United States  | 682        | \$1,085,399 | 480        | \$770,362   |            |
| Grand Total      | 2721       | \$4,354,080 | 2396       | \$3,852,530 |            |

### Adding a report

When you select the **“Create a new report”** option from the toolbar, the [newReport](#) event is triggered, followed by the [saveReport](#) event. To save this new report to the SQL database, use the [saveReport](#) event triggered later, and then follow the save report briefing in the preceding [topic](#).



|             | Grand Total           |
|-------------|-----------------------|
| Grand Total | No records to display |

### Limitations with respect to report manipulation

Below points need to be considered when saving the report to SQL Server database.

- **Data source:** Both raw data and aggregated data won't be saved and loaded from the database.
- **Hyperlinks:** Option to link external facts via pivot table cells won't be saved and loaded from the database.
- The pivot table should always load reports from the SQL database based on the data source that is currently bound to it.

In [this](#) GitHub repository, you can find our ASP.NET MVC Pivot Table sample and ASP.NET Core Web Application to save and load reports from SQL Server database.

### Events

#### FetchReport

The event [FetchReport](#) is triggered when dropdown list is clicked in the toolbar in-order to retrieve and populate saved reports. It has following parameter - **ReportName**. This event allows user to fetch the report names from local storage and populate the dropdown list.

#### LoadReport

The event [LoadReport](#) is triggered when a report is selected from the dropdown list in the toolbar. It has following parameters - **Report** and **ReportName**. This event allows user to load the selected report to the pivot table.

#### NewReport

The event [NewReport](#) is triggered when the new report icon is clicked in the toolbar. It has following parameter - **Report**. This event allows user to create new report and add to the report list.

*RenameReport*

The event [RenameReport](#) is triggered when rename report icon is clicked in the toolbar. It has following parameters - **Rename**, **Report** and **ReportName**. This event allows user to rename the selected report from the report list.

*RemoveReport*

The event [RemoveReport](#) is triggered when remove report icon is clicked in the toolbar. It has following parameters - **Report** and **ReportName**. This event allows user to remove the selected report from the report list.

*SaveReport*

The event [SaveReport](#) is triggered when save report icon is clicked in the toolbar. It has following parameters - **Report** and **ReportName**. This event allows user to save the altered report to the report list.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
"NumberFormatting", "FieldList"
}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport").RemoveReport("removeReport").NewReport("newReport").Render()
<style>
 #pivotview {
 width: 100%;
 height: 100%;
 }
 .e-tool-expand::before {
```

```

 content: '\e702';
 }
</style>
<script>
 function saveReport(args) {
 var reports = [];
 var isSaved = false;
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reports = JSON.parse(localStorage.pivotviewReports);
 }
 if (args.report && args.reportName && args.reportName !== '') {
 reports.map(function (item) {
 if (args.reportName === item.reportName) {
 item.report = args.report;
 isSaved = true;
 }
 });
 if (!isSaved) {
 reports.push(args);
 }
 localStorage.pivotviewReports = JSON.stringify(reports);
 }
 }
 function fetchReport(args) {
 var reportCollection = [];
 var reeportList = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 reeportList.push(item.reportName);
 });
 args.reportName = reeportList;
 }
 function loadReport(args) {
 var pivotObj =
 document.getElementById('pivotview').ej2_instances[0];
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 args.report = item.report;
 }
 });
 if (args.report) {
 pivotObj.dataSourceSettings =
 JSON.parse(args.report).dataSourceSettings;
 }
 }
 function removeReport(args) {
 var reportCollection = [];
 }

```

```

 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 for (var i = 0; i < reportCollection.length; i++) {
 if (reportCollection[i].reportName === args.reportName) {
 reportCollection.splice(i, 1);
 }
 }
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 localStorage.pivotviewReports =
 JSON.stringify(reportCollection);
 }
 }
 function renameReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 item.reportName = args.rename;
 }
 });
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 localStorage.pivotviewReports =
 JSON.stringify(reportCollection);
 }
 }
 function newReport() {
 var pivotObj =
 document.getElementById('pivotview').ej2_instances[0];
 pivotObj.setProperties({
 dataSourceSettings: {
 columns: [],
 rows: [],
 values: [],
 filters: []
 }
 }, false);
 }
}
</script>

```

### TOOLBAR.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                |            |             |            |             |            |         |
|----------------|------------|-------------|------------|-------------|------------|---------|
| Default report |            |             |            |             |            |         |
|                | FY 2015    |             | FY 2016    |             | FY 2017    |         |
|                | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Am |
| France         | 450        | \$714,955   | 526        | \$1,542,104 | 592        | \$2,    |
| Germany        | 440        | \$563,515   | 496        | \$1,772,104 | 372        | \$1,    |
| United States  | 546        | \$754,515   | 636        | \$2,263,104 | 731        | \$3,    |
| Grand Total    | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1695       | \$7,    |

<!-- markdownlint-disable MD009 -->

### ToolbarRender

The [ToolbarRender](#) event is triggered when the toolbar is rendered. It has the `customToolbar` parameter. This event helps to customize the built-in toolbar items and to [include new toolbar item\(s\)](#).

### CSSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("300").ShowToolbar(t
true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).EnableSorting
(true)
.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add(); })
.Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
.Columns(columns => { columns.Name("Year").Add();
columns.Name("Order_Source").Caption("Order Source").Add(); })
.Values(values =>
{
values.Name("In_Stock").Caption("In Stock").Add();
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
.Filters(filters =>
{
filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).DisplayOption(new PivotViewDisplayOption { View = View.Both
}).Toolbar(new List<string>
() { "Save", "Export", "FieldList"
}).SaveReport("saveReport").ToolbarRender("beforeToolbarRender").Render()
<style>
#pivotview {
width: 100%;
height: 100%;
}
.e-tool-expand::before {
content: '\e702';
}
</style>
<script>
function saveReport(args) {
```

```

 var reports = [];
 var isSaved = false;
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reports = JSON.parse(localStorage.pivotviewReports);
 }
 if (args.report && args.reportName && args.reportName !== '') {
 reports.map(function (item) {
 if (args.reportName === item.reportName) {
 item.report = args.report;
 isSaved = true;
 }
 });
 if (!isSaved) {
 reports.push(args);
 }
 localStorage.pivotviewReports = JSON.stringify(reports);
 }
 function beforeToolbarRender(args) {
 args.customToolbar.splice(2, 0, {
 prefixIcon: 'e-rename-report e-icons', tooltipText: 'Custom
Button',
 click: function (args) {
 // Here you can customize the click event for custom button
 },
 });
 args.customToolbar.splice(3, 0, {
 prefixIcon: 'e-tool-expand e-icons', tooltipText:
'Expand/Collapse',
 click: function (args) {
 var pivotTableObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotTableObj.dataSourceSettings.expandAll =
!pivotTableObj.dataSourceSettings.expandAll;
 },
 });
 args.customToolbar[0].align = "Left";
 args.customToolbar[1].align = "Center";
 args.customToolbar[2].align = "Right";
 }
 }
</script>

```

#### TOOLBARCUSTOMIZE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



|               | FY 2015    |             | FY 2016    |
|---------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 5          |
| Germany       | 440        | \$563,515   | 4          |
| United States | 546        | \$754,515   | 6          |
| Grand Total   | 1436       | \$2,032,985 | 16         |

### Before Export

The pivot table (or) pivot chart can be exported as a pdf, excel, csv etc., document using the toolbar options. And, you can customize the export settings for exporting document by using the [BeforeExport](#) event in the toolbar.

For example, you can add the header and footer for the pdf document by setting the **header** and **footer** properties for the **pdfExportProperties** in the [BeforeExport](#) event.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%).Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
```

```

"NumberFormatting", "FieldList"
}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport").RemoveReport("removeReport").NewReport("newReport").BeforeExport("beforeExport").Render()
<style>
 #pivotview {
 width: 100%;
 height: 100%;
 }
 .e-tool-expand::before {
 content: '\e702';
 }
</style>
<script>
 function saveReport(args) {
 var reports = [];
 var isSaved = false;
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reports = JSON.parse(localStorage.pivotviewReports);
 }
 if (args.report && args.reportName && args.reportName != '') {
 reports.map(function (item) {
 if (args.reportName === item.reportName) {
 item.report = args.report;
 isSaved = true;
 }
 });
 if (!isSaved) {
 reports.push(args);
 }
 localStorage.pivotviewReports = JSON.stringify(reports);
 }
 }
 function fetchReport(args) {
 var reportCollection = [];
 var reeportList = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 reeportList.push(item.reportName);
 });
 args.reportName = reeportList;
 }
 function loadReport(args) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 args.report = item.report;
 }
 });
 }

```

```

 }
 });
 if (args.report) {
 pivotObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
 }
}
function removeReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 for (var i = 0; i < reportCollection.length; i++) {
 if (reportCollection[i].reportName === args.reportName) {
 reportCollection.splice(i, 1);
 }
 }
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function renameReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 item.reportName = args.rename;
 }
 });
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function newReport() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.setProperties({
 dataSourceSettings: {
 columns: [],
 rows: [],
 values: [],
 filters: []
 }
 }, false);
}
function beforeExport(args) {
 args.pdfExportProperties = {
 header: {
 fromTop: 0,

```

```

 height: 130,
 contents: [
 {
 type: 'Text',
 value: "Pivot Table",
 position: { x: 0, y: 50 },
 }
]
 },
 footer: {
 contents: [
 {
 type: 'Text',
 value: "Thank You",
 style: { textBrushColor: '#FF0000', fontSize:
13, dashStyle: 'Solid', hAlign: 'Center' }
 }
]
 }
};
args.excelExportProperties = {
 header: {
 headerRows: 2,
 rows: [
 {
 cells: [
 {
 colSpan: 4,
 value: 'Pivot Table',
 style: {
 fontColor: '#C67878',
 fontSize: 20,
 hAlign: 'Center',
 bold: true,
 underline: true,
 }
 },
],
 },
],
 },
};
}
</script>

```

#### TOOLBAREXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |
|---------------|------------|-------------|------------|-------------|------------|-------------|------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 592        | \$2,903,308 |            |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 372        | \$1,634,808 |            |
| United States | 546        | \$754,515   | 636        | \$2,263,104 | 632        | \$3,041,448 |            |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1596       | \$7,579,564 | 1          |

### ActionBegin

The event [actionBegin](#) triggers when the UI actions such as switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc. that are present in toolbar UI begin. This allows user to identify the current action being performed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action began. The following are the UI actions and their names:

| Action | Action Name |

|-----|-----|

| New report | Add new report |

| Save report | Save current report |

| Save as report | Save as current report |

| Rename report | Rename current report |

| Remove report | Remove current report |

| Report change | Report change |

| Conditional Formatting | Open conditional formatting dialog |

| Number Formatting | Open number formatting dialog |

| Export menu | PDF export, Excel export, CSV export |

| Show Fieldlist | Open field list |

| Show Table | Show table view |

| Chart menu | Show chart view |

| Sub-totals menu | Hide sub-totals, Show row sub-totals, Show column sub-totals, Show sub-totals |

| Grand totals menu | Hide grand totals, Show row grand totals, Show column grand totals, Show grand totals |

- **cancel**: It allows user to restrict the current action.

In the below sample, toolbar UI actions such as add new report and save current report can be restricted by setting the **args.cancel** option to **true** in the **actionBegin** event.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolBar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolBar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
Syncfusion.EJ2.PivotView.View.Both }).ToolBar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
"NumberFormatting", "FieldList" }).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 if (args.actionName == 'Add new report' || args.actionName == 'Save
current report') {
 args.cancel = true;
 }
 }
</script>
```

### ACTIONBEGIN-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ActionComplete

The event [actionComplete](#) triggers when the UI actions such as switching between pivot table and pivot chart, changing chart types, conditional formatting, exporting, etc. that are present in toolbar UI, is completed. This allows user to identify the current UI actions being completed at runtime. It has the following parameters:

- **dataSourceSettings**: It holds the current data source settings such as input data source, rows, columns, values, filters, format settings and so on.
- **actionName**: It holds the name of the current action completed. The following are the UI actions and their names:

| Action                 | Action Name                                                                                |
|------------------------|--------------------------------------------------------------------------------------------|
| -----                  | -----                                                                                      |
| New report             | New report added                                                                           |
| Save report            | Report saved                                                                               |
| Save as report         | Report re-saved                                                                            |
| Rename report          | Report renamed                                                                             |
| Remove report          | Report removed                                                                             |
| Report change          | Report changed                                                                             |
| Conditional Formatting | Conditionally formatted                                                                    |
| Number Formatting      | Number formatted                                                                           |
| Export menu            | PDF exported, Excel exported, CSV exported                                                 |
| Show Fieldlist         | Field list closed                                                                          |
| Show Table             | Table view shown                                                                           |
| Sub-totals menu        | Sub-totals hidden, Row sub-totals shown, Column sub-totals shown, Sub-totals shown         |
| Grand totals menu      | Grand totals hidden, Row grand totals shown, Column grand totals shown, Grand totals shown |

- **actionInfo**: It holds the unique information about the current UI action. For example, while adding new report, the event argument contains information such as report name and the action name.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
```

```

{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
"NumberFormatting", "FieldList" }).ActionComplete("actionComplete").Render()
<script>
 function actionComplete(args) {
 if (args.actionName == 'New report added' || args.actionName ==
'Report saved') {
 // Triggers when the toolbar UI actions such as add new report
 and save current report icon are completed.
 }
 }
}
</script>

```

### ACTIONCOMPLETE-AGGREGATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### ActionFailure

The event [actionFailure](#) triggers when the current UI action fails to achieve the desired result. It has the following parameters:

- actionName:** It holds the name of the current action failed. The following are the UI actions and their names:

| Action      | Action Name         |
|-------------|---------------------|
| New report  | Add new report      |
| Save report | Save current report |



| Save as report | Save as current report |

| Rename report | Rename current report |

| Remove report | Remove current report |

| Report change | Report change |

| Conditional Formatting | Open conditional formatting dialog |

| Number Formatting | Open number formatting dialog |

| Export menu | PDF export, Excel export, CSV export |

| Show Fieldlist | Open field list |

| Show Table | Show table view |

| Chart menu | Show chart view |

| Sub-totals menu | Hide sub-totals, Show row sub-totals, Show column sub-totals, Show sub-totals |

| Grand totals menu | Hide grand totals, Show row grand totals, Show column grand totals, Show grand totals |

- **errorInfo**: It holds the error information of the current UI action.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolbar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolbar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
})
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
})
.Columns(columns =>
{
 columns.Name("Year").Add(); columns.Name("Quarter").Add();
})
.Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})
).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
"NumberFormatting", "FieldList" }).ActionFailure("actionFailure").Render()
```

```
<script>
 function actionFailure(args) {
 if (args.actionName == 'Add new report' || args.actionName == 'Save
current report') {
 // Triggers when the current UI action fails to achieve the
desired result.
 }
 }
}
</script>
```

### ACTIONFAILURE-AGGREGATION.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Tooltip

The tooltip can be enabled or disabled by setting the [showTooltip](#) property to **true**. By default, tooltip is enabled in the pivot table.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowTooltip(false).Render()
```

### TOOLTIP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

```
}
```

### Tooltip Template

User can design their own tooltip by setting the property [TooltipTemplate](#) with own HTML elements. The property accepts both HTML string and ID attribute. The following place holders are available to display its dynamic values inside the HTML elements.

`${rowHeaders}` – Row headers of the selected value cell.

`${columnHeaders}` – Column headers of the selected value cell.

`${rowFields}` – Row fields of the selected value cell.

`${columnFields}` – Column fields of the selected value cell.

`${valueField}` – Field name of the selected value cell.

`${aggregateType}` – Aggregate type of the selected value cell.

`${value}` - Formatted value of the selected value cell.

The tooltip customization is common for both pivot table and pivot chart or it can be done individually as well. To customize the pivot table tooltip, the above procedure needs to be followed. To customize the pivot chart tooltip alone use `Template` property of tooltip under [ChartSettings](#).

In the below sample, the pivot table and pivot chart shows customized tooltip layouts.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").TooltipTemplate("#Template").ShowToolBar(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add(); })
.Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
.Columns(columns => { columns.Name("Year").Add();
columns.Name("Order_Source").Caption("Order Source").Add(); })
.Values(values =>
{
values.Name("In_Stock").Caption("In Stock").Add();
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})
.Filters(filters =>
{
filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).DisplayOption(new PivotViewDisplayOption { View = View.Both
}).ToolBar(new List<string>
() { "Grid", "Chart" }).ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column).Tooltip(tooltip =>
tooltip.Template("${aggregateType} of ${valueField}:
${value}"))).Render()
```

```

<style>
 #pivotview {
 width: 100%;
 height: 100%;
 }
 .e-tool-expand::before {
 content: '\e702';
 }
 .wrap {
 border: 3px solid #27b1f0;
 background-color: #4d4d4d;
 width: auto;
 color: #FFFFFF;
 padding: 5px;
 font-size: 12px;
 }
 .pivotTooltipValue {
 font-style: italic;
 }
 .pivotTooltipHeader {
 color: aqua;
 font-weight: bold;
 width: 100px;
 }
</style>
<script id="Template" type="text/x-template">
 <div class='wrap'>
 <div>
 Row Headers :
 >
 ${columnHeaders}${rowHeaders}
 </div>
 <div>
 Row Fields :

 ${rowFields}
 </div>
 <div>
 Column Headers
:
 ${columnHeaders}
 </div>
 <div>
 Column Fields
:
 ${columnFields}
 </div>
 <div>
 Value Field
:
 ${valueField}
 </div>
 <div>
 Value :

```

```

 ${value}
 </div>
</script>

```

### TOOLTIP-TEMPLATE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

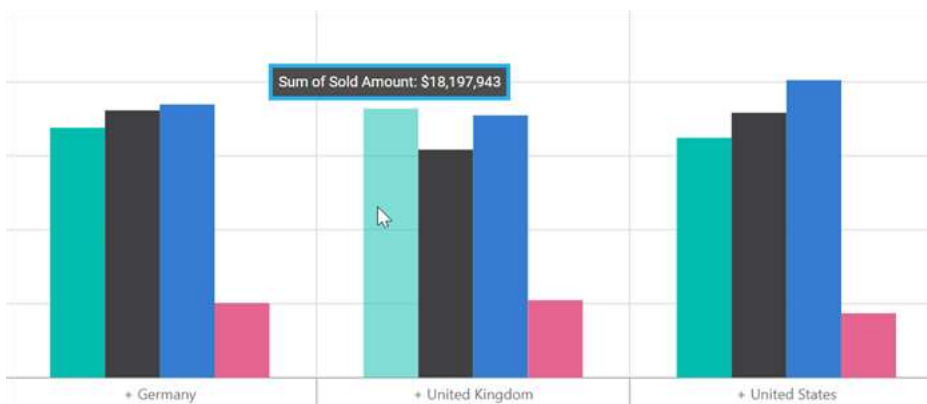
<!-- markdownlint-disable MD012 -->

|                | FY 2015    |              | FY 2016    |              | FY 2017    |
|----------------|------------|--------------|------------|--------------|------------|
|                | Units Sold | Sold Amount  | Units Sold | Sold Amount  | Units Sold |
| France         | 46939      | \$19,097,074 | 164058     | \$70,994,281 | 48         |
| Germany        | 44154      | \$16,422,943 | 164058     | \$70,994,281 | 47         |
| United Kingdom | 25457      | \$18,197,943 | 164058     | \$70,994,281 | 25         |
| United States  | 47724      | \$16,422,943 | 164058     | \$70,994,281 | 46         |
| Grand Total    | 164274     | \$70,422,943 | 164058     | \$70,994,281 | 167        |

<br/>

<br/>

<br/>



## Style and Appearance

### Hiding Axis

The visibility of row, column, value and filter axis in Field List and Grouping Bar can be changed using custom CSS setting. To do so, refer the code sample below:

#### CSHTML

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).ShowFieldList(true).Render()
<style>
 #PivotTable .e-group-columns {
 display: none;
 }
 #PivotTable .e-group-filters {
 height: 71px !important;
 }
 #PivotTable_PivotFieldList_Wrapper .e-field-list-columns{
 display: none;
 }
 #PivotTable_PivotFieldList_Wrapper .e-field-list-values{
 margin-top: 0px;
 height: 338px;
 }
 .e-pivotfieldlist-wrapper .e-values {
 height: 310px !important;
 }
 /* Hiding row axis in grouping bar */
 /* #PivotView .e-group-rows {
 display: none;
 } */
 /* Hiding row axis in field list */
 /* .e-pivotfieldlist-wrapper .e-field-list-rows {
 display: none;
 } */
 /* Hiding value axis in grouping bar */
 /* #PivotView .e-group-values {
 display: none;
 } */
 /* Hiding value axis in field list */
 /* .e-pivotfieldlist-wrapper .e-field-list-values {
 display: none;
 } */
 /* Hiding filter axis in grouping bar */
 /* #PivotView .e-group-filters {
```

```

 display: none;
 } */
 /* Hiding filter axis in field list */
 /* .e-pivotfieldlist-wrapper .e-field-list-filters {
 display: none;
 } */
</style>

```

### AXIS.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Text Alignment

The alignment of text inside row headers, column headers, value cells and summary cells can be changed using custom CSS setting. To do so, refer the code sample below:

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource => dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()
<style>
 .e-pivotview .e-valuescontent {
 text-align: center !important;
 }

 /* Column headers */
 /*.e-pivotview .e-columnsheader {
 text-align: center !important;
 }
 //Rows Headers
 .e-pivotview .e-rowsheader {
 text-align: center !important;
 }*/
 /* Summary Cells */
 /* .e-pivotview .e-summary {
 text-align: center !important;
 }*/
</style>

```

**TEXT-ALIGN.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**Customize header, value and summary cell style**

The elements in pivot table like header cell, value cell and summary cell style can be customized using built-in CSS names. To do so, refer the code sample below:

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).ShowGroupingBar(true).ShowFieldList(true).Render()
<style>
 .e-pivotview .e-headercell {
 background-color: thistle !important;
 }
 .e-pivotview .e-rowsheader {
 background-color: skyblue !important;
 }
 .e-pivotview .e-summary:not(.e-gtot) {
 background-color: pink !important;
 }
 .e-pivotview .e-gtot {
 background-color: greenYellow !important;
 }
</style>
```

**AXIS.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```



|                | FY 2016 |             | FY 2017    |             | FY 2018    |             | Grand Total |              |
|----------------|---------|-------------|------------|-------------|------------|-------------|-------------|--------------|
|                | Sold    | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  | Sold Amount  |
| France         | 609     | \$983,317   | 703        | \$1,140,998 | 68         | \$108,402   | 2109        | \$3,392,816  |
| Germany        | 667     | \$1,067,220 | 579        | \$845,569   | 130        | \$211,903   | 1304        | \$3,070,364  |
| United Kingdom | 640     | \$1,031,631 | 657        | \$1,041,051 | 161        | \$263,367   | 2240        | \$3,601,197  |
| United States  | 480     | \$770,362   | 644        | \$1,022,552 | 232        | \$366,358   | 2098        | \$3,344,670  |
| Grand Total    | 2396    | \$3,852,530 | 2583       | \$4,150,169 | 591        | \$952,029   | 8291        | \$13,308,807 |

## Printing and Exporting

### Print in ASP.NET MVC Pivot Table Component

The rendered pivot table can be printed directly from the browser by invoking the `print` method from the grid's instance. The below sample code illustrates the print option being invoked by an external button click.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("print").Content("Print").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).Render()
<script>
 var pivotObj;
 document.getElementById('print').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.grid.print();
 }
</script>
```

#### PRINTTABLE.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

Similarly, to print the pivot chart, use the `print` method from the chart's instance. The below sample code illustrates the print option being invoked by an external button click.

To display the pivot chart, set the [PivotViewDisplayOption](#) property to either **Chart** or **Both**.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().Button("print").Content("Print").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DisplayOption(new PivotViewDisplayOption { View = View.Chart
}).ChartSettings(chartSettings => chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()
<script>
 var pivotObj;
 document.getElementById('print').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.chart.print();
 }
</script>
```

### PRINTCHART.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Excel Export in ASP.NET MVC Pivot Table Component

The Excel export allows Pivot Table data to be exported as Excel document. To enable Excel export in the pivot table, set the [AllowExcelExport](#) property in [PivotView](#) class to **true**. Once the API is set, user needs to call the `excelExport` method for exporting on external button click.

**Note:** The pivot table component can be exported to Excel format using options available in the toolbar. For more details [refer](#) here.

**CSHTML**

```
@Html.EJS().Button("excel").Content("Export To
Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})) .AllowExcelExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.excelExport();
 }
</script>
```

**EXPORT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*Multiple pivot table exporting*

The Excel export provides an option to export multiple pivot table data in the same Excel file.

*Same WorkSheet*

The Excel export provides support to export multiple pivot tables in same sheet. To export in same sheet, define `multipleExport.type` as `AppendToSheet` in `excelExportProperties`. It has an option to provide blank rows between pivot tables and these blank row(s) count can be defined using the `multipleExport.blankRows` property.

**Note:** By default, `multipleExport.blankRows` value is 5 between pivot table's within the same sheet.

**CSHTML**

```
@Html.EJS().Button("excel").Content("Export To
Excel").IsPrimary(true).Render()
```

```

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()

@Html.EJS().PivotView("PivotGrid2").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Columns(columns =>
{
columns.Name("Country").Add(); columns.Name("Products").Add();
}).Rows(rows =>
{
rows.Name("Year").Caption("Year").Add(); rows.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()
<script>
var pivotObj; var pivotObj2;
document.getElementById('excel').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
pivotObj2 = document.getElementById('PivotGrid2').ej2_instances[0];
var excelExportProperties = {
multipleExport: { type: 'AppendToSheet', blankRows: 2 }
};
var firstGridExport =
pivotObj.grid.excelExport(excelExportProperties, true);
firstGridExport.then(function (fData) {
pivotObj2.excelExport(excelExportProperties, false, fData);
});
}
</script>

```

**SAMESHEETEXPORT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**New WorkSheet**

Excel export provides support to export multiple pivot tables into new sheets. To export in new sheets, define `multipleExport.type` as `NewSheet` in `excelExportProperties`.

**CSHTML**

```
@Html.EJS().Button("excel").Content("Export To
Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()

@Html.EJS().PivotView("PivotGrid2").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Columns(columns =>
{
 columns.Name("Country").Add(); columns.Name("Products").Add();
}).Rows(rows =>
{
 rows.Name("Year").Caption("Year").Add(); rows.Name("Quarter").Add();
```

```

 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).AllowExcelExport(true).Render()
<script>
 var pivotObj; var pivotObj2;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj2 = document.getElementById('PivotGrid2').ej2_instances[0];
 var excelExportProperties = {
 multipleExport: { type: 'NewSheet' }
 };
 var firstGridExport =
 pivotObj.grid.excelExport(excelExportProperties, true);
 firstGridExport.then(function (fData) {
 pivotObj2.excelExport(excelExportProperties, false, fData);
 });
 }
</script>

```

### NEWSHEETEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### *Changing the pivot table style while exporting*

The Excel export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **excelExportProperties** object and pass it as a parameter to the **excelExport** method.

**Note:** By default, material theme is applied to exported Excel document.

### CSHTML

```

@Html.EJS().Button("excel").Content("Export To Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{

```

```

 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).AllowExcelExport(true).Render()
<script>
 var pivotObj; var pivotObj2;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var excelExportProperties = {
 theme: {
 header: { fontName: 'Segoe UI', fontColor: '#666666' },
 record: { fontName: 'Segoe UI', fontColor: '#666666' },
 caption: { fontName: 'Segoe UI', fontColor: '#666666' }
 }
 };
 pivotObj.excelExport(excelExportProperties);
 }
</script>

```

### THEMEEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Add header and footer while exporting

The Excel export provides an option to include header and footer content for the excel document before exporting. In-order to add header and footer, define **header** and **footer** properties in **excelExportProperties** object and pass it as a parameter to the **excelExport** method.

### CSHTML

```

@Html.EJS().Button("excel").Content("Export To Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{

```

```

 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).AllowExcelExport(true).Render()
<script>
 var pivotObj; var pivotObj2;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var excelExportProperties = {
 header: {
 headerRows: 2,
 rows: [
 { cells: [{ colSpan: 4, value: "Pivot Grid", style: {
fontColor: '#C67878', fontSize: 20, hAlign: 'Center', bold: true, underline:
true } }] }
],
 },
 footer: {
 footerRows: 4,
 rows: [
 { cells: [{ colSpan: 4, value: "Thank you for your
business!", style: { hAlign: 'Center', bold: true } }] },
 { cells: [{ colSpan: 4, value: "!Visit Again!", style: {
hAlign: 'Center', bold: true } }] }
],
 },
 };
 pivotObj.excelExport(excelExportProperties);
 }
</script>

```

### THEMEEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Changing the file name while exporting

The Excel export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **excelExportProperties** object and pass it as a parameter to the **excelExport** method.

### CSHTML

```

@Html.EJS().Button("excel").Content("Export To
Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>

```



```

dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()
<script>
 var pivotObj; var pivotObj2;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var excelExportProperties = {
 fileName: 'sample.xlsx'
 };
 pivotObj.excelExport(excelExportProperties);
 }
</script>

```

### FILENAME.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### CSV Export

Also, the Excel export allows pivot table data to be exported in **CSV** file format. To export pivot table in **CSV** file format, you need to use the **csvExport** method.

### CSHTML

```

@Html.EJS().Button("excel").Content("Export To Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{

```

```

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()
<script>
 var pivotObj; var pivotObj2;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.csvExport();
 }
</script>

```

### CSVEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Virtual Scroll Data

You can export the pivot table virtual scroll data as Excel/CSV document by using PivotEngine export without any performance degradation. To enable PivotEngine export in the pivot table, set the [AllowExcelExport](#) as true. You need to use the `exportToExcel` method for PivotEngine export.

**Note:** PivotEngine export will be performed while enabling virtual scrolling by default.

### Virtual Scroll Data Excel Export

#### CSHTML

```

@Html.EJS().Button("excel").Content("Export To Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}

```

```

 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).AllowExcelExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.excelExportModule.exportToExcel('Excel');
 }
</script>

```

**EXPORT.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

## Virtual Scroll Data CSV Export

**CSHTML**

```

@Html.EJS().Button("excel").Content("Export To Excel").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()
<script>
 var pivotObj; var pivotObj2;
 document.getElementById('excel').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];

```

```

 var excelExportProperties = {
 fileName: 'csvexport.csv',
 };
 pivotObj.csvExport(excelExportProperties);
 }
</script>

```

### CSVEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Export all pages

The pivot engine exports the entire virtual data of the pivot table (i.e. the data that contains all of the records used to render the complete pivot table) as an Excel/CSV document. To export just the current viewport of the pivot table, set the [exportAllPages](#) property to **false**. To use the pivot engine export, add the [ExcelExport](#) module into the pivot table.

**Note:** By default, the pivot engine export will be performed while virtual scrolling is enabled.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
ExportAllPages(false).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(true)
e)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Sold").Add(); rows.Name("Amount").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Country").Caption("Units Sold").Add();
values.Name("Products").Caption("Sold Amount").Add();
})).AllowExcelExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.excelExport();
 }
</script>

```

**EXPORTALLPAGES.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*Events**ExcelQueryCellInfo*

The event **ExcelQueryCellInfo** triggers while framing each row and value cell during Excel export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- **value** - It holds the cell value.
- **column** - It holds column information for the current cell.
- **data** - It holds the entire row data across the current cell.
- **style** - It holds the style properties for the cell.

**CSHTML**

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.ExcelQueryCellInfo("excelQueryCell").Render()
<script>
 function excelQueryCell(args) {
 }
</script>
```

**EXCELQUERYCELL.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ExcelHeaderQueryCellInfo

The event **ExcelHeaderQueryCellInfo** triggers on framing each header cell during Excel export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- **cell** - It holds the current cell information.
- **style** - It holds the style properties for the cell.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.ExcelHeaderCellInfo("excelHeaderCell").Render()
<script>
 function excelHeaderCell(args) {
 }
</script>
```

### EXCELHEADER.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### ExportComplete

The event [ExportComplete](#) is triggered after the pivot table data has been exported to an Excel/CSV document. You can use this event to acquire blob stream data for further customization and processing at your end by passing the `isBlob` parameter as **true** when using the `excelExport` method. It has the following parameters:

- `type` - It holds the current export type such as PDF, Excel, and CSV.
- `promise` - It holds the promise object for blob data.

### CSHTML

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
e)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Sold").Add(); rows.Name("Amount").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Country").Caption("Units Sold").Add();
values.Name("Products").Caption("Sold Amount").Add();
})).AllowExcelExport(true).ExportComplete("exportComplete").Render()
<script>
var pivotObj;
document.getElementById('pdf').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
pivotObj.excelExport({}, false, null, true);
}
function exportComplete(args) {
if (args.promise !== null) {
args.promise.then((e: { blobData: Blob }) => {
console.log(e.blobData);
});
}
}
}
</script>
```

### BLOB-EXPORT.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
}
```

```
return View();
}
```

*See Also*

- [PDF Exporting](#)

### PDF Export in ASP.NET MVC Pivot Table Component

PDF export allows exporting pivot table data as PDF document. To enable PDF export in the pivot table, set the [AllowPdfExport](#) as true. You need to use the `pdfExport` method for PDF exporting.

#### CSHTML

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
var pivotObj;
document.getElementById('pdf').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
pivotObj.pdfExport();
}
</script>
```

#### EXPORT.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```



*Multiple pivot table exporting*

PDF export provides an option for exporting multiple pivot tables to same file. In this exported document, each pivot table will be exported to new page of document in same file.

**CSHTML**

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()

@Html.EJS().PivotView("PivotGrid2").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Columns(columns =>
{
columns.Name("Country").Add(); columns.Name("Products").Add();
}).Rows(rows =>
{
rows.Name("Year").Caption("Year").Add(); rows.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
var pivotObj; var pivotObj2;
document.getElementById('pdf').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
pivotObj2 = document.getElementById('PivotGrid2').ej2_instances[0];
var firstGridPdfExport = pivotObj.grid.pdfExport({}, true);
firstGridPdfExport.then(function (pdfData) {
pivotObj2.pdfExport({}, false, pdfData);
```

```

 });
}
</script>

```

### MULTIPLEEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### *Export table and chart into the same document*

When the [PivotViewDisplayOption](#) is set to **Both**, you can export both the table and the chart into the same PDF document. To achieve this, use the `pdfExport` method and set the `exportBothTableAndChart` parameter to **true**.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DisplayOption(new
PivotViewDisplayOption { View = View.Both }).DataSourceSettings(dataSource
=>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render().ChartSettings(chartSettings =>
chartSettings.ChartSeries(chartSeries =>
chartSeries.Type(ChartSeriesType.Column))).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExport(null, false, null, false, true);
 }
</script>

```

### EXPORTTABLEANDCHART.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

#### *Customization during PDF export*

PDF export provides option to customize mapping of pivot table to the exported PDF document.

#### *To add header and footer while exporting*

You can customize text, page number, line, page size and changing orientation in header and footer of the exported document.

#### *To add a text in header/footer*

You can add text either in header or footer of the exported PDF document like in the below code example.

```
`javascript
var pdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'Text',
 value: "Northwind Traders",
 position: { x: 0, y: 50 },
 style: { textBrushColor: '#000000', fontSize: 13 }
 },
]
 }
},
```

#### *To draw a line in header/footer*

You can add line either in header or footer of the exported PDF document like in the below code example.

Supported line styles:

- dash
- dot
- dashdot

- dashdotdot
- solid

```
`javascript
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Line',
style: { penColor: '#000080', penSize: 2, dashStyle: 'Solid' },
points: { x1: 0, y1: 4, x2: 685, y2: 4 }
}
]
}
}
```

#### [Add page number in header/footer](#)

You can add page number either in header or footer of exported PDF document like in the below code example.

Supported page number types:

- LowerLatin - a, b, c,
- UpperLatin - A, B, C,
- LowerRoman - i, ii, iii,
- UpperRoman - I, II, III,
- Number - 1,2,3.

```
`javascript
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'PageNumber',
```

```

pageNumberType: 'Arabic',
format: 'Page { $current } of { $total }', //optional
position: { x: 0, y: 25 },
style: { textBrushColor: '#ffff80', fontSize: 15, hAlign: 'Center' }
}
]
}
}
,

```

The below code illustrates the PDF export customization options.

### **CSSHTML**

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
var pivotObj;
document.getElementById('pdf').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Text',
value: "Pivot Table",
position: { x: 0, y: 50 },
style: { textBrushColor: '#000000', fontSize: 13,
dashStyle: 'Solid', hAlign: 'Center' }
}
}
}
}

```

```

],
 footer: {
 fromBottom: 160,
 height: 150,
 contents: [
 {
 type: 'PageNumber',
 pageNumberType: 'Arabic',
 format: 'Page {$current} of {$total}',
 position: { x: 0, y: 25 },
 style: { textBrushColor: '#02007a', fontSize: 15 }
 }
]
 }
};
pivotObj.pdfExport(pdfExportProperties);
}
</script>

```

### THEMEEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### Add an image in header/footer

You can add image (Base64 string) either in header or footer of the exported PDF document like in the below code example.

```

`javascript
var pdfExportProperties = {
header: {
fromTop: 0,
height: 130,
contents: [
{
type: 'Image',
src: image,
position: { x: 20, y: 10 },
size: { height: 100, width: 100 },
}
]
}

```

$$\left. \begin{array}{l} \} \\ \} \\ , \end{array} \right\}$$

The below code illustrates the PDF export customization options.

CSHTML

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).AllowPdfExport(true).Render()
```

<script>

```
var pivotObj;
```

```
var image =
```

[illegible]

zxZdy7i/eb5Hfnx8WbAg3U9RLTSJLEtzk3muaGOdixyJeilhaHVpGsyDZ2QjJEhruOTwOY7ja5OJ  
T3EG4fOrSXJBZRH7GI79Pskzw/g0XdV6eTd3OPou54Scj6xYVqutKnzj23Kmhcf4fvQfOrXs9tDN  
kNdei6fVOIAB2j1AiDLC4DKjyyU1zSXyIMsLgMqPLJTXNJNF4hnFPk8P2VKRGaURAAAAB6qN44ke  
Uwuuh5T1UbxxI8phddDC6jKHTE5542cMN3c7R+sp0MOeeNnDDd3O0frKV6xe8dwJU+pDUAAWIiAA  
AAttKlBxbi53TUQypJbbIW3i3FzumohnOtXwy+3ybYdssKACqk0ifEGypmTjR6vTfhZmWiOWJHhq  
quiQlVc6uz+Fzf909KbqaIWTI7vywmxvhKnQoTWxfrRZVNxH+VWeRfR4F4sy+Gg290YVL6ikt1Vv  
3T8csC42PbyLdDUrwX8/nniRgD9clzXK1zVa5q5lRUzKipxKfhRS2gAAA+8hKTM/OwpOUhLfjxnb  
FjE41/wCE41XiQ+LGue9rGNc971RGtamdVVFaiJxqTNh3ajaFJ/G5trXVKO35/H8E3w7BP+V8v7j  
r2NZMlpt5CaGprXBPYu7mc607RZQxZS6XLqT97jI2bbsvbtLSXYqRJmJmdMRs313eRPupxJ/yqmb  
APrdPBHTxpFglzU1HzaaZ8z1ket6qAABjWCIMsLgMqPLJTXNJfIgywuAyo8slNc0k0XiGcU+Tw/Z  
UpGAC5EAAAHAqo3jiR5TC66HlPVRvHEjymF10MLqModMTnnjZww3dztH6ynQw5542cMN3c7R+spX  
rF7x3Alt6kNQABYiIAAAC22QtfFuLndNRDKkltshbeLcXO6aiGc6lFDL7fJth2ywoAKqTQAADUL6  
suXrbXTsjsJepIm67wNjZuJ3p+9+OfiiGclZiTmokrNwXwI8Ncz2PTMqL/wB4+MsaYK77YkbiLUB  
G+hm4aZoMw1N1voXyt9H4ZipW70aZV3z0+h+9Nzvwrv34lj3i3XU10U+lmO9P9ftMCCR4Eznvrt  
In6LPukqhB+DiJutcm62I3/M1eNP+qbbhhafx+M2tVKFnlIa55eG5P0rk/WX7qf7r6E3aHSWbUVV  
T1Zrbnb792KqW+proYIM+q3t3Xb+BlsL7S+KsZXKnCzTD0zy0Jyfo2r+uv3l4vInpXckeA+uWfQR  
UECQxak1riuKnzetrJKyVZP+JgAATSKAAACKMrCQn6lgvPylMkJufmXTcqrYMrAdFiKiRmqgo1q  
KuZE3SVwbIZM1I1+C3mFS9LjnB3GXlodcvqiY7A7jLy0OuX1RMdg6Pg6/bTvJ9TR1dMTnB3GXlod  
cvqiY7A7jLy0OuX1RMdg6PgdtO8n1HV0xOcHcZeWhly+qJjsHqpFnXi2rStNwfcjWpMw1VVPmwiI  
mzTdx5h0VAW2neT6jQ6YgoZjHallzWLF1TMratfIEWqRnw4sGmR3se1XLmVrkaqKnpQvmDn0dWt  
K5XI195tezLS45wdx15aHXL6omOwO4y8tDrl9UTHYoj40h207yfUldXTE5wdx15aHXL6omOwO4y8  
tDrl9UTHYoj4HbTvJ9R1dMTnB3GXlodcvqiY7BabIrpVVPn11+DVqVUKbFiVRHsZOSr4LnN+Bhpn  
RHoiqmdFTOnkJ5BHqbTDPGsatuPTIclb7wADmG4AAAAA8NbpFPrMokrUZZseGjtk3OqorV8qKm6  
h7IUNkKE2FCY1kNjUa1rUzI1E8CIh/QPCRMR6vREvXWu/QelkcrUYq6E3AAHs8AAAAAAaijKynp6  
nYLT81Tp6bkZhs3KokaWjuhPRFjNRURzVRcyoSuRB1hCb1R5ZKa5pJo0vqGcU+Ty/ZUP73X3dpdc  
frWP2x3X3dpdcfrWP2zCguGQ3AgXma7r7u0uuPlrH7Y7r7u0uuPlrH7ZhQMhuAvM13X3dpdcfrWP  
2z00i7btdVpJrrsuJzVmYaKi1WOqKmzTcX55rh6qN44keUwuuhhWNU1BFOMJQRGS6LolsWbql5a5  
67LwIVUjthwoVSjMYxqOXmJWo7MiehC+pzxs4Ybu52j9ZSv2MiLI6/Alt6kMV3X3dpdcfrWP2x3  
X3dpdcfrWP2zCgsOQ3Ai3ma7r7u0uuPlrH7Y7r7u0uuPlrH7ZhQMhuAvM13X3dpdcfrWP2y0+RVV  
KpVLKr8Wq1SfqERlVRrHzcy+M5rfGya5kV6qqJnVVzekp8W2yFt4txc7pqIZz7Ua1KZbkW+TbCv6  
yUsX5iYlralny0xGgPWcaihRFYqpsH7mdP3EV/K1Wek1DpT/AO5J+NG9eV5czVxKJD4N0qle20F  
RFXUh9K6OxsdRIqpvU9nytVvOtQ6U/wDuPlaredah0p/9zxgrmfk8y8zu5qPypPZ8rVbZrUOlP8  
A7j5Wq3nWodKf/c8YGfk8y8xmo/KnIzNv1SqPr9NY+pzzmunIKOa6YeqKivTOipnJ6K9W7vipnLY  
OsaWFL/0Me50UuUt+lCm9KGNbJHcl2hQAC6FWAAAAABEGWfWGVHlKprmkvkQZYXAZUeWSmuaSaL  
xDOKfJ4fsqJABciAAAAD1UbxxI8phddDynqo3jiR5TC66GF1GUOmJzzxs4Ybu52j9ZToYc88bOG  
G7udo/WUrli947gSp9SGoAAsREAAABbbIW3i3FzumohlSS22QtfFuLndNRDODavhl9vk2w7ZJuNG  
9eV5czVxCJCW8aN68ry5mriESHwLpZ/MV4IfTujngk4qAAVo7oAAB7rd3xUzlsHWNLClerd3xUz1  
SHWNLCn0HoT3UvFCmdKu8j4KAAXCqgAAAAAIgywuAyo8slNc0l8iDLC4DKjyyU1zSTReIZxT5PD  
9lSkYALkQAAAAeqjeOJHlMLroeU9VG8cSPKYXXQwuoyh0xOeeNnDDd3O0frKdDDnnjZww3dztH6y  
lesXvHcCVPqQ1AAFiIgAAALbZC28W4ud01EMqSW2yFt4txc7pqIZzrV8Mvt8m2HbJNxo3ryvLmau  
IRIS3jRvXleXM1cQiQ+BdLP5ivBD6d0c8EnFQACTHdAAAPdbu+Kmctg6xpYUrbu+Kmctg6xpYU+  
g9Ce6l4oUzpV3kfBQAC7lUAAAAABruItUq+rWjW7WYk1Dk40SHEc6Wejh52ORyZlVFTwp5DYjV  
cV7117AsuYuaakI09CgRYUNYMJ6NcqvejEXOu5uZ85siR6vTI17jC3XaSONq7h19uuLpcP3Y2ruH  
X264ulw/dmC22FE0OqnSoY22FE0OqnSoZ1Mi0fXmhpviM7tXcOvt1xdLh+7G1dw6+3XF0uH7swW2  
womh1U6VDG2womh1U6VDGRaPrzQXxGd2ruHX264ulw/dn0lsmPD2XmYUdk9cKvhPa9uebh5s6LnT  
/AA/Qa9tsKJodVOLQz+5fKsokWPDhJZ9URXvRuf4zD3M65jGRaPrzQXxFiyH7nydrFuG46hXZ6cr  
rZqfmHzEZIUzDRIocudcyLDXMn+pMBBF6ZSLJtm7apb0alqjMRKdMul3RWTDEa9W8aIu6hCpUnVy  
5jWbH5N36j7bV3DrH7dcXS4fuxtXcOvt1xdLh+7MFtsKJodVOLQxtsKJodVOLQydkWj680Nd8Rndq  
7h19uuLpcP3Y2ruHX264ulw/dmC22FE0OqnSoY22FE0OqnSoYyLR9eaC+Izuldw6+3XF0uH7skHC  
zDqhYc0qcplBjT8WDNzHxiIs3Fa9yO2KN3FRrdzM1CidthRNDqp0qGSjgtiZJ4mUioVGTPuZTmyU  
yku5kaI16uVWI7Omibi3TRUNrEjXO35J6asd/6TabmoUnX5BknOvjNhsipFRYtKrc6IqcaLufOU17  
vZ2/+3qP81vZP3GPEGVw3taBXpumx6hDjTjJVIUGiJHirmPdss68XzP8AciXbYUTQ6qdKhniF0fi  
rlzzoUcuJOitOembkMkVEJZ72dv8A7eo/zW9kd7O3/wBvUf5reyRNtsKJodVOLQxtsKJodVOLQzz  
/AAjT/wBun0NnblX/AFVJZ72dv/t6j/Nb2R3s7f8A29R/mt7JE22womh1U6VDG2womh1U6VDH8I0  
/9un0Hb1X/VU16Sw6oUpOQJqHGnlfAitiNR0VqpnaqKmf5voNwK/UDKgo1Wr9NpLLTqUJ8/OQZVs  
R0zDVGLEejEcvoRXZywJsjsplnfzGjL8DRNWylSosjsq4AA2GkAAAAAAEQ5YHAXU+VymvaS8RD1  
gcBdT5XKa9pJovEM4p8niTYUpEAC5EAAAHA3p3jCW/jM6yHwPvTvGet/GZ1kMLqModNDnvjpwYxb



```

zpF/M6EHPfHThku3nSL+ZxRf713D7kqfZNMABYiIAAAC2eQrvLuTnVupYVMLZ5Cu8u5OdW6lhZrV
8Mvt8m2HbMplucEtP57g6mMU4Lj5bnBLT+e4OpjFODfK+H91MzbQAB0jSAAAZzDvhEtfnuS/qGHR
85wYd8Ilr89yX9Qw6PlftraZ7kqn1KAACQkAAAAAAiHLA4C6nyuU17SXiIcsDgLqfK5TXtJNF4h
nFPk8SbClIgAXIgAAAA+908YS38ZnWQ+B96d4wlv4zOshhdRlDpoc98dOGS7edIv5nQg5746cMl2
86RfzK9Yveu4fclT7JpgALERAAAWzyFd5dyc6t1LCphbPIV3l3Jzq3UsOdavh19vk2w7ZlMtZgl
p/PcHUxinBcfLc4Jafz3B1MYpwYsnw/upmbaAA0kaQAAD0Yd8Ilr89yX9Qw6PnODDvhEtfnuS/qG
HR8r9tbTPclU+pQADiEgAAAAAEQ5YHAXU+VymvaS8RnlOUGsXJhDP0qg0+NUJ6JMyz2QIWbZORs
ZquXdVE3ERVJFIqJOxVxT5PD9lSiAn77zmKOhFU9n2h3nMUdCKp7PtFu6xF505oQsl2BogN77zmK
OhFU9n2h3nMUdCKp7PtDrEXnTmgyXYGiH3p3jCW/jM6yG6d5zFHQiqez7R9ZHB/E9k7Ae+yqojWx
Wqq/R7iIqfeMLURXbac0CNdgX7Oe+OnDJdvOkX8zoQUoxewsxEquKNy1Om2jUZqTmahEiQIzNhsX
tVdxUzuznAsd7WSOVy3aPuSZ0VU0ENg3vvOYo6EVT2faHecxR0Iqns+0WDrEXnTmhGyXYGiA3vvO
Yo6EVT2faHecxR0Iqns+0OsRedOaDJdgaIWzyFd5dyc6t1LCB+85ijoRVPZ9osjkg2pcdqWrXZa5
KPM0uNMVFsSEyNsc72fBNTOmZV40VDn2nNG6nVGURdW/1NsLVR2o+WW5wS0/nuDqYxTgu5lZ23Xr
ow2kqdbtLj1KbZVoUZ0KDm2SMSFFRXbqpuZ3J+JV7vOYo6EVT2faMWXLG2nuc5E0rvEzVV2hDRAb
33nMUdCKp7PtDvOYo6EVT2faOj1iLzpzQ1ZLsDRAb33nMUdCKp7PtDvOYo6EVT2faHWIvOnNBkuw
MBh3wiWvz3Jf1DDo+UWsjCXEqSve352as2pwpeXqspGjRHLdZMY2MxznL87wIiKpek4VsSMe5uSq
KSYEVEW8AA4xvAAAP/9k=";
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var pdfExportProperties = {
 header: {
 fromTop: 0,
 height: 130,
 contents: [
 {
 type: 'Image',
 src: image,
 position: { x: 20, y: 10 },
 size: { height: 100, width: 100 },
 }
]
 }
 };
 pivotObj.pdfExport(pdfExportProperties);
 }
</script>

```

### THEMEEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Changing the file name while exporting

The PDF export provides an option to change file name of the document before exporting. In-order to change the file name, define **fileName** property in **pdfExportProperties** object and pass it as a parameter to the **pdfExport** method.

### CSHTML

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
```

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
var pivotObj;
document.getElementById('pdf').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
var pdfExportProperties = {
fileName: 'sample.pdf'
};
pivotObj.pdfExport(pdfExportProperties);
}
</script>
```

### FILENAME.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

### Changing page orientation while exporting

The PDF export provides an option to change page orientation of the document before exporting. In order to change the page orientation, define **pageOrientation** property in **pdfExportProperties** object and pass it as a parameter to the **pdfExport** method. By default, the page orientation will be in **Portrait** and it can be changed to **Landscape** based on user requirement.

### CSHTML

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
```

```

{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var pdfExportProperties = {
 pageOrientation: 'Landscape'
 };
 pivotObj.pdfExport(pdfExportProperties);
 }
</script>

```

### PAGELAYOUT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### Changing page size while exporting

The PDF export provides an option to change page size of the document before exporting. In-order to change the page size, define **pageSize** property in **pdfExportProperties** object and pass it as a parameter to the **pdfExport** method.

**Supported page sizes are:** Letter, Note, Legal, A0, A1, A2, A3, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5, ArchA, ArchB, ArchC, ArchD, Arche, Flsa, HalfLetter, Letter11x17, Ledger.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{

```

```

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var pdfExportProperties = {
 pageSize: 'Letter'
 };
 pivotObj.pdfExport(pdfExportProperties);
 }
</script>

```

### PAGESIZE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Changing document width and height while exporting

Before exporting, you can change the height and width of the PDF document. To achieve this, use the **height** and **width** properties in the [BeforeExport](#) event.

This option is only available if [EnableVirtualization](#) is set to **true**. In addition, the **VirtualScroll** and **PDFExport** modules must be injected into the pivot table.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>

```

```

{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).AllowPdfExport(true).Render().BeforeExport("beforeExport").Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExport();
 }
 function beforeExport(args: BeforeExportEventArgs) {
 args.width = pivotObj.element.offsetWidth;
 args.height = pivotObj.element.offsetHeight;
 }
</script>

```

### EXPORTING-CUSTOMIZATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### Customize the table column count while exporting

Before exporting, you can split and export the pivot table columns on each page of the PDF document by using the **columnSize** property in the [BeforeExport](#) event.

This option is only available if [EnableVirtualization](#) is set to **true**. In addition, the [VirtualScroll](#) and [PDFExport](#) modules must be injected into the pivot table.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>

```

```

{
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render().BeforeExport("beforeExport").Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExport();
 }
 function beforeExport(args: BeforeExportEventArgs) {
 args.columnSize = 6;
 }
</script>

```

### EXPORTING-COLUMNCUSTOMIZATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

#### Changing the table's column width and row height while exporting

You can change the column width and row height in the PDF document during the pivot table export by using the [OnPdfCellRender](#) event. Within this event, the `args.column.width` property allows you to change the width of specific columns.

As shown in the code example below, the “Unit Sold” column under “FY 2015” is changed to a width of 60 pixels.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
}

```

```

 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).AllowPdfExport(true).OnPdfCellRender("onPdfCellRender").Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExport();
 }
 function onPdfCellRender(args) {
 if (args.pivotCell && args.pivotCell.valueSort &&
 args.pivotCell.valueSort.levelName === 'FY 2015.Units Sold') {
 args.column.width = 60
 }
 }
</script>

```

### EXPORTING-COLUMNWIDTHCUSTOMIZATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

Similarly, you can change the height of specific rows in the PDF document by using the `args.cell.height` property in the [OnPdfCellRender](#) event.

As shown in the code example below, the “Mountain Bikes” row under “France” is changed to a height of 30 pixels.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).DrilledMembers(drilledmembers =>
{
 drilledmembers.Name("Country").Items(new string[] { "France" }).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
}

```

```

 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).AllowPdfExport(true).Render().OnPdfCellRender("onPdfCellRender").Render(
)
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExport();
 }
 function onPdfCellRender(args) {
 if (args.pivotCell && args.pivotCell.valueSort &&
 args.pivotCell.valueSort.levelName === 'France.Mountain Bikes') {
 args.cell.height = 30
 }
 }
</script>

```

### EXPORTING-ROWHEIGHTCUSTOMIZATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

This option is only available if [EnableVirtualization](#) is set to **true**.

#### *Changing the pivot table style while exporting*

The PDF export provides an option to change colors for headers, caption and records in pivot table before exporting. In-order to apply colors, define **theme** settings in **pdfExportProperties** object and pass it as a parameter to the **pdfExport** method.

**Note:** By default, material theme is applied to exported PDF document.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{

```



```

 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).AllowPdfExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var pdfExportProperties = {
 theme: {
 header: {
 fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true, borders: { color: '#64FA50', lineStyle: 'Thin' }
 },
 record: {
 fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
 },
 caption: {
 fontColor: '#64FA50', fontName: 'Calibri', fontSize: 17,
bold: true
 }
 }
 };
 pivotObj.pdfExport(pdfExportProperties);
 }
</script>

```

### THEMEEXPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

<!-- markdownlint-disable MD009 -->

#### Changing default font while exporting

By default, the pivot table uses "Helvetica" font in the exported document. But it can be changed using the **theme** property in **pdfExportProperties**.

The available built-in fonts are,

- Helvetica
- TimesRoman
- Courier
- Symbol
- ZapfDingbats

```
`javascript
var pdfExportProperties = {
 theme: {
 header: {font: new PdfStandardFont(PdfFontFamily.TimesRoman, 11, PdfFontStyle.Bold) },
 caption: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 9) },
 record: { font: new PdfStandardFont(PdfFontFamily.TimesRoman, 10) }
 }
}
```

#### Adding custom font while exporting

In addition to existing built-in fonts, custom fonts can also be used. The custom font should be in **Base64** format and mention it in **PdfTrueTypeFont** class. In the following example, we have used **Advent Pro** font family that supports **Hungarian** language

#### CSHTML

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).AllowPdfExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 var pdfExportProperties = {
 theme: {
 header: {font: new PdfTrueTypeFont(base64AlgeriaFont, 11)
 },
 caption: { font: new PdfTrueTypeFont(base64AlgeriaFont, 9)
 },
 record: { font: new PdfTrueTypeFont(base64AlgeriaFont, 10) }
 }
 }
```

```

 };
 pivotObj.pdfExport (pdfExportProperties);
}
</script>

```

### **NON-ENGLISH-EXPORT.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

**Note:** The non-English alphabets can also be exported properly by setting its appropriate font.

#### *Virtual Scroll Data*

You can export the pivot table virtual scroll data as PDF document by using PivotEngine export without any performance degradation. To enable PivotEngine export in the pivot table, set the `allowPdfExport` as true. You need to use the `exportToPDF` method for PivotEngine export.

**Note:** PivotEngine export will be performed while enabling virtual scrolling by default

### **CSHTML**

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExportModule.exportToPDF();
 }
</script>

```

**EXPORT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**Repeat row headers**

Repeat row headers on each page can be achieved using PivotEngine export option. To disable repeat row headers, you need to set **allowRepeatHeader** to **false** in beforeExport event. You need to use the **exportToPDF** method for PivotEngine export.

**Note:** By default, repeat row headers is enabled in the PivotEngine export.

**CSHTML**

```
@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).AllowPdfExport(true).BeforeExport("beforeExport").Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExportModule.exportToPDF();
 }
 function beforeExport(args: BeforeExportEventArgs) {
 args.allowRepeatHeader = false;
 }
</script>
```

**EXPORT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
```

```

 ViewBag.DataSource = data;
 return View();
}

```

### Export all pages

The pivot engine exports the entire virtual data of the pivot table (i.e. the data that contains all of the records used to render the complete pivot table) as a PDF document. To export just the current viewport of the pivot table, set the [exportAllPages](#) property to **false**. To use the pivot engine export, add the **PDFExport** module into the pivot table.

**Note:** By default, the pivot engine export will be performed while virtual scrolling is enabled.

### CSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
ExportAllPages(false).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(true)
)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Sold").Add(); rows.Name("Amount").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Country").Caption("Units Sold").Add();
values.Name("Products").Caption("Sold Amount").Add();
})).AllowPdfExport(true).Render()
<script>
 var pivotObj;
 document.getElementById('pdf').onclick = function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.pdfExport();
 }
</script>

```

### EXPORTALLPAGES.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Events

#### PdfQueryCellInfo

The event **PdfQueryCellInfo** triggers on framing each row and value cell during PDF export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- **value** - It holds the cell value.
- **column** - It holds column information for the current cell.
- **data** - It holds the entire row data across the current cell.
- **style** - It holds the style properties for the cell.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.PdfQueryCellInfo("pdfqueryCell").Render()
<script>
function pdfqueryCell(args) {
}
</script>
```

### PDFQUERYCELL.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

#### PdfHeaderQueryCellInfo

The event **PdfHeaderQueryCellInfo** triggers on framing each column header cell during PDF export. It allows the user to customize the cell value, style etc. of the current cell. It has the following parameters:

- **cell** - It holds the current rendering cell information.
- **style** - It holds the style properties for the cell.

### CSSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.PdfHeaderCellInfo("pdfHeaderCell")).Render()
<script>
function pdfHeaderCell(args) {
}
</script>
```

### PDFHEADER.CS

```
public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}
```

### ExportComplete

The event [ExportComplete](#) is triggered after the pivot table data has been exported to an PDF document. You can use this event to acquire blob stream data for further customization and processing at your end by passing the **isBlob** parameter as **true** when using the **pdfExport** method. It has the following parameters:

- **type** - It holds the current export type such as PDF, Excel, and CSV.
- **promise** - It holds the promise object for blob data.

### CSSHTML

```

@Html.EJS().Button("pdf").Content("Pdf Export").IsPrimary(true).Render()
@Html.EJS().PivotView("PivotView").Height("300").EnableVirtualization(true).
DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(true)
e)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Sold").Add(); rows.Name("Amount").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Country").Caption("Units Sold").Add();
values.Name("Products").Caption("Sold Amount").Add();
})).AllowPdfExport(true).ExportComplete("exportComplete").Render()
<script>
var pivotObj;
document.getElementById('pdf').onclick = function () {
pivotObj = document.getElementById('PivotView').ej2_instances[0];
pivotObj.pdfExport({}, false, null, true);
}
function exportComplete(args) {
if (args.promise !== null) {
args.promise.then((e: { blobData: Blob }) => {
console.log(e.blobData);
});
}
}
}
</script>

```

### BLOB-EXPORT.CS

```

public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```

See Also

- [Excel Exporting](#)

### Globalization

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number ([Internationalization](#)) & adding culture specific customization and translation to the text ([Localization](#)).



### Internationalization

Internationalization library provides support for formatting and parsing the number, date, and time by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data.

By default, all the Essential JS 2 component are specific to English culture ('en-US'). If you want to go with the different culture other than English, follow the below steps.

- Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this [link](#).

,

```
npm install cldr-data --save
```

,

Once the package installed, you can find the culture specific JSON data under the location `/scripts/cldr-data`.

- Now use the [loadCultureFiles](#) method to load the culture specific CLDR JSON data.

In ASP.NET MVC refer the culture files directly from `/scripts/cldr-data` location. In ASP.NET Core refer the culture files directly from `/wwwroot/scripts/cldr-data` location as like the below code examples for both ASP.NET Core and MVC

```
`sh
```

```
function loadCultureFiles(de) {
```

```
var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
```

```
var loader = ej.base.loadCldr;
```

```
var loadCulture = function (prop) {
```

```
var val, ajax;
```

```
<!--For ASP.NET MVC -->
```

```
ajax = new ej.base.Ajax(location.origin + '/../scripts/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
```

```
<!--For ASP.NET Core-->
```

```
ajax = new ej.base.Ajax(location.origin + '/../wwwroot/scripts/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
```

```
ajax.onSuccess = function (value) {
```

```
val = value;
```

```
};
```

```
ajax.send();
```

```
loader(JSON.parse(val));
```

```
};
for (var prop = 0; prop < files.length; prop++) {
loadCulture(prop);
}
}
,
```

- Before changing to a culture other than English, ensure that locale text for the concerned culture is loaded through load method of L10n class.

```
`sh
var L10n = ej.base.L10n;
L10n.load({
"de": {
'pivotview': {
'grandTotal': 'Gesamtsumme',
'total': 'Insgesamt',
'value': 'Wert',
'noValue': 'Kein Wert',
'row': 'Zeile',
'column': 'Spalte',
'collapse': 'Zusammenbruch',
'expand': 'Erweitern',
},
}
});
,
```

- Set the culture by using the locale property.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
```

```

formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Render()
<script>
 var pivotObj;
 var L10n = ej.base.L10n;
 L10n.load({
 "fr-CH": {
 "pivotview": {
 "grandTotal": "Grand Total",
 "total": "Total",
 "value": "Valeur",
 "noValue": "Pas de valeur",
 "row": "Rangée",
 "column": "La colonne",
 "collapse": "Effondrement",
 "expand": "L'expansion"
 },
 "pivotfieldlist": {
 "fieldList": "Field List",
 "dropRowPrompt": "Drop row ici",
 "dropColPrompt": "Drop column here",
 "dropValPrompt": "Valeur liste déroulante ici",
 "dropFilterPrompt": "Filtre goutte ici",
 "addPrompt": "Ajouter un champ ici",
 "centerHeader": "Faire glisser les champs entre les zones
ci-dessous :",
 "add": "Ajouter",
 "drag": "Faites glisser",
 "filter": "Filter",
 "filtered": "Filtered",
 "sort": "Trier",
 "remove": "Déposer",
 "filters": "Filters",
 "rows": "Lignes",
 "columns": "Colonnes",
 "values": "Valeurs",
 "error": "Error",
 "dropAction": "Champ calculé ne peut pas être place dans
toute autre région, à l'exception de l'axe des valeurs.",
 "search": "Recherchez",
 "close": "Fermer",
 "cancel": "Annuler",
 "delete": "Supprimer",
 "alert": "Alert",

```

```

 "warning": "Attention",
 "ok": "OK",
 "allFields": "Tous les domaines",
 "noMatches": "Pas de correspondance"
 }
}
});
document.addEventListener('DOMContentLoaded', function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 loadCultureFiles('fr-CH');
 pivotObj.locale = 'fr-CH';
});
function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
 if (name === 'ar') {
 files.push('numberingSystems.json');
 }
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;
 if (name === 'ar' && prop === files.length - 1) {
 ajax = new ej.base.Ajax(location.origin +
'../../scripts/cldr-data/supplemental/' + files[prop], 'GET', false);
 } else {
 ajax = new ej.base.Ajax(location.origin +
'../../scripts/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
 };
 for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
 }
}
</script>

```

### INTERNATIONALIZATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

**Note:** \* By default, locale value is en-US. If you want to change the en-US culture to a different culture, you have to change the locale accordingly.

<!-- markdownlint-disable MD009 -->

*Decimal separators*

The decimal separators of pivot table values varies based on the culture applied to the component. The culture can be set by calling the method [setCulture](#) with appropriate culture string as its parameter.

The following example demonstrates the decimal separators in **Deutsch** culture.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C2").Currency("EUR").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Locale("de-DE").Render()
<script>
var pivotObj;
var L10n = ej.base.L10n;
L10n.load({
 'de-DE': {
 'pivotview': {
 'grandTotal': 'Gesamtsumme',
 'total': 'Insgesamt',
 'value': 'Wert',
 'noValue': 'Kein Wert',
 'row': 'Zeile',
 'column': 'Spalte',
 'collapse': 'Zusammenbruch',
 'expand': 'Erweitern'
 },
 'pivotfieldlist': {
 'fieldList': 'Feld Liste',
 'dropRowPrompt': 'Drop Reihe hier',
 'dropColPrompt': 'Drop column Hier',
 'dropValPrompt': 'Drop wert hier',
 'dropFilterPrompt': 'Drop Filter Hier',
 'addPrompt': 'Feld hinzufügen',
 'centerHeader': 'Ziehen Sie die Felder zwischen den Bereichen
unten:',
 'add': 'Hinzufügen',
 'drag': 'Ziehen',
 'filters': 'Filter',
 'rows': 'Zeilen',
 'columns': 'Spalten',
 'values': 'Werte',
```

```

 'error': 'Fehler',
 'dropAction': 'Berechnetes Feld nicht in jeder anderen Region
außer Wert Achse sein.',
 'search': 'Suche',
 'close': 'Schließen',
 'cancel': 'Abbrechen',
 'delete': 'Löschen',
 'alert': 'Warnung',
 'warning': 'Warnung',
 'ok': 'OK',
 'allFields': 'Alle Felder',
 'noMatches': 'Keine Treffer'
 }
}
});
document.addEventListener('DOMContentLoaded', function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 loadCultureFiles('de-DE');
});
function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
 if (name === 'ar') {
 files.push('numberingSystems.json');
 }
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;
 if (name === 'ar' && prop === files.length - 1) {
 ajax = new ej.base.Ajax(location.origin +
'../../scripts/cldr-data/supplemental/' + files[prop], 'GET', false);
 } else {
 ajax = new ej.base.Ajax(location.origin +
'../../scripts/cldr-data/main/' + name + '/' + files[prop], 'GET', false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
 ej.base.setCulture('de');
 ej.base.setCurrencyCode('EUR');
 };
 for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
 }
}
</script>

```

## LOCALE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

}

|                    |                  |                 |            |    |
|--------------------|------------------|-----------------|------------|----|
| Sum of Units Sold  | Drop filter here |                 |            |    |
| Sum of Sold Amount | Production Year  |                 | Quarter    |    |
| Country            | FY 2015          |                 | FY 2016    |    |
| Products           | Units Sold       | Sold Amount     | Units Sold | \$ |
| France             | 46939            | 19.097.073,84 € | 44513      | 1  |
| Germany            | 44154            | 16.903.257,27 € | 47910      | 1  |
| United Kingdom     | 25457            | 18.197.943,25 € | 23944      | 1  |
| United States      | 47724            | 16.224.668,54 € | 47691      | 1  |
| Gesamtsumme        | 164274           | 70.422.942,9... | 164058     | 7  |

Localization

The [Localization](#) library allows you to localize default text content of the Pivot Table. The pivot Table component has static text on some features (like drop area text, pivot field list title, etc...) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the `locale` value and translation object.

The following list of properties and its values are used in the pivot table.

- Locale keywords |Text
- staticFieldList | Pivot Field List
- fieldList | Field List
- dropFilterPrompt | Drop filter here
- dropColPrompt | Drop column here
- dropRowPrompt | Drop row here
- dropValPrompt | Drop value here
- addPrompt | Add field here
- adaptiveFieldHeader | Choose field
- centerHeader | Drag fields between axes below:
- add | add
- drag | Drag
- filter | Filter
- filtered | Filtered
- sort | Sort

remove | Remove

filters | Filters

rows | Rows

columns | Columns

values | Values

calculatedField | Calculated Field

createCalculatedField | Create Calculated Field

fieldName | Enter the field name

error | Error

invalidFormula | Invalid formula.

dropText | Example: ("Sum(*OrderCount*)" + "Sum(*InStock*)") \* 250

dropTextMobile | Add fields and edit formula here.

dropAction | Calculated field cannot be place in any other region except value axis.

search | Search

close | Close

cancel | Cancel

delete | Delete

alert | Alert

warning | Warning

ok | OK

sum | Sum

average | Average

count | Count

min | Min

max | Max

allFields | All Fields

formula | Formula

fieldExist | A field already exists in this name. Enter a different name.

confirmText | A calculation field already exists in this name. Do you want to replace it?

noMatches | No matches

format | Summaries values by

edit | Edit

clear | Clear



formulaField | Drag and drop fields to formula

dragField | Drag field to formula

### Loading Translations

To load translation object in an application, use [load](#) function of the [L10n](#) class.

The following example demonstrates the Pivot Table in **Deutsch** culture.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Locale("fr-CH").Render()
<script>
var L10n = ej.base.L10n;
L10n.load({
 "fr-CH": {
 "pivotview": {
 "grandTotal": "Grand Total",
 "total": "Total",
 "value": "Valeur",
 "noValue": "Pas de valeur",
 "row": "Rangée",
 "column": "La colonne",
 "collapse": "Effondrement",
 "expand": "L'expansion"
 },
 "pivotfieldlist": {
 "fieldList": "Field List",
 "dropRowPrompt": "Drop row ici",
 "dropColPrompt": "Drop column here",
 "dropValPrompt": "Valeur liste déroulante ici",
 "dropFilterPrompt": "Filtre goutte ici",
 "addPrompt": "Ajouter un champ ici",
 "centerHeader": "Faire glisser les champs entre les zones
ci-dessous :",
 "add": "Ajouter",
 "drag": "Faites glisser",
 "filter": "Filter",
```

```

 "filtered": "Filtered",
 "sort": "Trier",
 "remove": "Déposer",
 "filters": "Filters",
 "rows": "Lignes",
 "columns": "Colonnes",
 "values": "Valeurs",
 "error": "Error",
 "dropAction": "Champ calculé ne peut pas être place dans
toute autre région, à l'exception de l'axe des valeurs.",
 "search": "Recherchez",
 "close": "Fermer",
 "cancel": "Annuler",
 "delete": "Supprimer",
 "alert": "Alert",
 "warning": "Attention",
 "ok": "OK",
 "allFields": "Tous les domaines",
 "noMatches": "Pas de correspondance"
 }
}
});
</script>

```

### LOCALIZATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Right-to-left (RTL)

RTL provides an option to switch the text direction and layout of the Pivot Table component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Pivot Table, set the `enableRtl` property to **true**.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSet
tings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(fal
se)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).Mini
mumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{

```

```

 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).ShowFieldList(true).Locale("ar").EnableRtl(true).Render()
<script>
 var L10n = ej.base.L10n;
 L10n.load({
 "ar": {
 "pivotview": {
 "grandTotal": "المجموع الكلي",
 "total": "المجموع",
 "value": "القيمة",
 "noValue": "لا قيمة لها",
 "row": "صف",
 "column": "العمود",
 "collapse": "الانهيار",
 "expand": "توسيع"
 },
 "pivotfieldlist": {
 "fieldList": "قائمة الحقول",
 "dropRowPrompt": "تراجع الخلاف هنا",
 "dropColPrompt": "انخفاض العمود هنا",
 "dropValPrompt": "انخفاض قيمة هنا",
 "dropFilterPrompt": "انخفاض هنا عامل التصفية",
 "addPrompt": "إضافة حقل هنا",
 "centerHeader": "اسحب المجالات بين المناطق الموضحة أدناه",
 "add": "إضافة",
 "drag": "اسحب",
 "filter": "الفلتر",
 "filtered": "تصفية",
 "sort": "النوع",
 "remove": "قم بإزالة",
 "filters": "عوامل التصفية",
 "rows": "الصفوف",
 "columns": "الاعمدة",
 "values": "قيم",
 "error": "خطأ",
 "search": "البحث",
 "close": "قريب",
 "cancel": "الغاء",
 "delete": "احذف",
 "alert": "حالة تاهب قصوى",
 "warning": "تحذير",
 "ok": "موافق",
 "allFields": "جميع الحقول",
 "noMatches": "لا مباريات"
 }
 }
 });
 document.addEventListener('DOMContentLoaded', function () {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 ej.base.setStyleAttribute(pivotObj.fieldlistModule.element, {
 width: ej.base.formatUnit(pivotObj.width)
 });
 });

```

```
});
</script>
```

## RTL.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

## Accessibility in Pivotview component

The pivot table component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the pivot table component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

```
</style>
```

```
<div> - All
features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

### WAI-ARIA attributes

[WAI-ARIA](#) (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components. The following ARIA attributes are used in the pivot table component:

| Attributes | Purpose |

| --- | --- |

| **role=grid** | Attribute added to identify the grid component element within the pivot table element. |

| **role=region** | Attribute added to identify the chart component element within the pivot table element. |

| **role=button** | This attribute is added to the pager navigation buttons as well as the buttons in the dialog popup such as field list, calculated field, member editor, conditional formatting of pivot table component to indicate that it is a clickable element. |

| **role=table** | This attribute is added to each conditional formatting style container element to denote it as a table. |

| **role=tableitems** | This attribute is added to the container element that appears inside the number formatting popup to indicate it as a table. |

| **aria-disabled** | The buttons within the dialog popups, such as field list, calculated field and member editor, will be disabled based on their usability. To indicate its disabled state, we will add this attribute with the values **true**. By default, the attribute value is set to **false**. |

| **aria-label** | This attribute is added to label elements that are placed inside the pager, member editor popup, and calculated field popup to identify them as label elements. |

| **aria-selected** | This attribute is added to the selected treeview item in the calculated field popup with the value as **true** to denote that it is a selected element. |

| **aria-colspan** | This attribute is added to the **th** elements in the **e-table**, which represent the column span value. |

| **aria-rowspan** | This attribute is added to the **th** elements in the **e-table**, which represent the row span value. |

| **data-type** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It represents the aggregate type for the specified field. |

| **data-caption** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It represents the caption for the specified field. |

| **data-basefield** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It denotes the base field for the specified field, which is used to display the values for aggregation types such as **DifferenceFrom**, **PercentageOfDifferenceFrom**, and **PercentageOfParentTotal**. |

| **data-baseitem** | This attribute is added to the treeview item in the calculated field popup, as well as the buttons in the grouping bar and field list. It denotes the base item for the specified field, which is used to display the values for aggregation types such as **DifferenceFrom**, **PercentageOfDifferenceFrom**, and **PercentageOfParentTotal**. |

| **data-field** | This attribute is added to the treeview item in the calculated field popup. It denotes the name of the specified field. |

| **data-membertype** | This attribute is added to the treeview item in the calculated field popup. It denotes the member type of the selected OLAP calculated field. |

| **data-hierarchy** | This attribute is added to the treeview item in the calculated field popup. It denotes the parent hierarchy unique name of the selected OLAP calculated field. |

| **data-formula** | This attribute is added to the treeview item in the calculated field popup. It denotes the formula used for the specified calculated field. |

| **data-formatString** | This attribute is added to the treeview item in the calculated field popup. It denotes the format string used for the specified calculated field. |

| **data-customformatstring** | This attribute is added to the treeview item in the calculated field popup. It denotes the custom format string used for the specified calculated field. |

### Keyboard interaction

The pivot table component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Message component.

| **Press** | **To do this** |

| --- | --- |

| **Tab / Shift + Tab** | **To focus the close icon in the message.** |

| **Enter / Space** | **Closes the focused close icon's message.** |

### Pivot Table

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Moves the cell focus right side. If no cells are focused, it moves to the next active element in the browser page.** |

| **Shift + Tab** | **Moves the cell focus left side. If no cells are focused, it moves to the previous active element in the browser page.** |

| DownArrow | Moves the cell focus downwards. If the selection is enabled in the pivot table, then it will move downwards to select next row or column or individual cell. |

| UpArrow | Moves the cell focus upwards. If the selection is enabled in the pivot table, then it will move upwards to select previous row or column or individual cell. |

| LeftArrow | Moves the cell focus left side. If the selection is enabled in the pivot table, then it will move left side to select previous row or column or individual cell. |

| RightArrow | Moves the cell focus right side. If the selection is enabled in the pivot table, then it will move right side to select next row or column or individual cell. |

| Shift + DownArrow | Extends the cell selection downwards. |

| Shift + UpArrow | Extends the cell selection selection upwards. |

| Shift + LeftArrow | Extends the cell selection to the left side. |

| Shift + RightArrow | Extends the cell selection to the right side. |

| Ctrl + A | Selects all cells. |

| Esc | Deselects all cells. If the current active element is a context menu, then the context menu popup will be closed. |

| Home | Goes to the first cell in the current row. |

| End | Goes to the last cell in the current row. |

| Ctrl + Home | Goes to the first cell in the table. |

| Ctrl + End | Goes to the last cell in the table. |

| Enter | If the current cell is an expand/collapse cell, it performs expand/collapse operation (drill operation). If the current row/column header is in value sort state, it performs value sorting. If the current cell is in selection state, it moves to the next row, column or individual cell. If drill-through or editing is enabled in the pivot table, the drill-through dialog will be opened based on the selected value cell. If the current active element is a context menu popup, menu selection will be performed. |

| Shift + Enter | If value sorting is enabled in the pivot table and the current cell is a header with respect to its value axis, it performs value sorting to either ascending or descending order. If the current cell is in selection state, it moves to the previous row, column or individual cell. |

| Ctrl + Enter | If hyperlink is enabled in the current cell, it performs hyperlink selection. |

| Shift + F10 or Menu | If context menu is enabled in the pivot table, the context menu popup will be opened in the current cell. |

#### Field List

| Press | To do this |

| --- | --- |

| Shift + Ctrl + F | If the popup field list is enabled in either the pivot table or the pivot chart, the field list dialog will be opened. |

| Tab | Moves to the next active element in the field list. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the field list. If no active elements present, it moves to the previous active element in the browser page. |

| Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will open to perform filtering. |

| Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field. |

| Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Enter | Performs the selection operation of the current active element. If the current active element is a field's button and it has a dropdown icon, the aggregation menu will open to perform calculations using aggregation options to the selected value field. |

| Delete | If the current active element is a field's button, the selected field will be removed from the current report. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. |

| RightArrow | If the current active element is a tree node, it expands the current node. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Esc or Escape | Closes the popup field list dialog. |

#### Grouping Bar

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element (field's button) in the grouping bar. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element (field's button) in the grouping bar. If no active elements present, it moves to the previous active element in the browser page. |

| Shift + F | If the current active element is a field's button and if it has a filter icon, the filter dialog will be opened to perform filtering. |

| Shift + S | If the current active element is a field's button and if it has a sort icon, the sorting will be performed to the selected field. |



| Shift + E | If the current active element is a calculated field's button and if it has an edit icon, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Enter | Performs the selection operation of the current active element. If the current active element is a field's button and if it has a dropdown icon, the aggregation menu will be opened to perform calculations using aggregation options to the selected value field. |

| Delete | If the current active element is a field's button, the selected field will be removed from the current report. |

| DownArrow | If the current active element is a dropdown list, the next item will be selected. |

| UpArrow | If the current active element is a dropdown list, the previous item will be selected. |

| Home | If the current active element is a dropdown list, the first item will be selected. |

| End | If the current active element is a dropdown list, the last item will be selected. |

| Alt + Down | If the current active element is a dropdown list, the popup will be opened. |

| Alt + Down | If the current active element is a dropdown list, the popup will be closed. |

| Esc or Escape | Closes the dropdown list.

#### *Filter Dialog*

| Press | To do this |

| --- | --- |

| Shift + F | If the current active element is a field's button and if it has a filter icon in either the field list or grouping bar UI, the filter dialog will be opened to perform filtering. |

| Tab | Moves to the next active element in the filter dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the filter dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. If the current active element is a tab, it moves focus to the previous tab element. |

| RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tab, it moves focus to the next tab element. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Space | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| Alt + Down | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be opened. |

| Alt + Up | If the current active element is a DropDownList or DatePicker or DateTimePicker, the popup will be closed. |

| Enter | Performs the selection operation of the current active element. If the current active element is a tab, the current tab element will be selected. If the current active element is a tree node, the current node will be either checked or unchecked. If the current active element is DropDownList, the focus item will be selected, and the popup list will close when it is open. Otherwise, toggles the popup list. |

| Esc or Escape | Closes the filter dialog. |

#### *Calculated Field Dialog*

| Press | To do this |

| --- | --- |

| Shift + E | If the current active element is a field's button and if it has an edit icon in either the field list or grouping bar UI, the calculated field dialog will be opened to perform editing the selected calculated field. |

| Tab | Moves to the next active element in the calculated field dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the calculated field dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a tree node, it moves to the next node. |

| UpArrow | If the current active element is a tree node, it moves to the previous node. |

| LeftArrow | If the current active element is a tree node, it collapses the current node. |

| RightArrow | If the current active element is a tree node, it expands the current node. If the current active element is a tree node and has a menu icon, the aggregation menu will be opened to select appropriate aggregation type to the selected field. |

| Home | If the current active element is a tree node, it goes to the first node. |

| End | If the current active element is a tree node, it goes to the last node. |

| Enter | Performs the selection operation of the current active element. If the current active element is a tree node, it copies the selected field name/formula to the formula text area to perform calculations. |

| Esc or Escape | Closes the calculated field dialog. |

#### *Formatting Dialog*

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element in the formatting dialog. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the formatting dialog. If no active elements present, it moves to the previous active element in the browser page. |

| DownArrow | If the current active element is a DropDownList, the next item will be selected. |

| UpArrow | If the current active element is a DropDownList, the previous item will be selected. |

| Home | If the current active element is a DropDownList, the first item will be selected. |

| End | If the current active element is a DropDownList, the last item will be selected. |

| Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be opened. |

| Alt + Down | If the current active element is a DropDownList or ColorPicker, the popup will be closed. |

| Enter | Performs the selection operation of the current active element. |

| Esc or Escape | Closes the formatting dialog. |

#### Toolbar

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active option in the toolbar. If no active elements present, it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active option in the toolbar. If no active elements present, it moves to the previous active element in the browser page. |

| Enter | Performs the selection operation of the current active element. |

#### Drill-Through Dialog

| Press | To do this |

| --- | --- |

| Tab | Moves to the next active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to right side. If no active elements present, then it moves to the next active element in the browser page. |

| Shift + Tab | Moves to the previous active element in the drill-through dialog. If the current active element is a Grid cell, it moves the cell focus to left side, If no active elements present, then it moves to the previous active element in the browser page. |

| DownArrow | Moves the row/cell focus downwards. |

| UpArrow | Moves the row/cell focus upwards. |

| LeftArrow | Moves the cell focus left side. |

- | **RightArrow** | Moves the cell focus right side. |
- | **Home** | Goes to the first cell in the current row. |
- | **End** | Goes to the last cell in the current row. |
- | **Ctrl + Home** | Goes to the first cell in the table. |
- | **Ctrl + End** | Goes to the last cell in the table. |
- | **Enter** | Performs the selection operation of the current active element. |
- | **Esc or Escape** | If the cell is in selected state, the it deselects all rows/cells. If the row/cell is in edit state, it cancels the current entries in the row/cell. If the current active element is not a row/cell, it closes the drill-through dialog. |
- | **F2** | Initiate editing a row/cell in the data grid. |
- | **Insert** | Adds a new row/cell in the data grid. |
- | **Delete** | Removes the selected row in the data grid. |

Some commonly used applicable key combinations and their relative functionalities in all dialogs are listed below.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves to the next active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the next active element in the browser page. |

| **Shift + Tab** | Moves to the previous active element in the dialog. If either no active elements present in the dialog or an overlay is not present in the dialog, it moves to the previous active element in the browser page. |

| **Space** | If the current active element is a tree node or a checkbox element, it will be either checked or unchecked. |

| **Enter** | When the Dialog button or any input (except text area) is in focus state, when pressing the Enter key, the click event associated with the primary button or button will be triggered. The Enter key will not be worked, when the dialog content contains any text area with initial focus. |

| **Esc or Escape** | Closes the dialog. |

### Ensuring accessibility

The pivot table component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the pivot table component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the pivot table component with accessibility tools.

### CSHTML

```

@Html.EJS().PivotView("pivotview").Width("100%").Height("450").ShowFieldList
(true).ShowGroupingBar(true).AllowExcelExport(true)
).AllowConditionalFormatting(true).AllowNumberFormatting(true).AllowPdfExport
(true).AllowGrouping(true).ShowToolbar(true).AllowCalculatedField(true)
).ShowFieldList(true).AllowDeferLayoutUpdate(true).SaveReport("saveReport").
LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport")
).RemoveReport("removeReport").NewReport("newReport").ToolbarRender("beforeT
oolbarRender").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource(

(IEnumerable<object>)ViewBag.Data).ExpandAll(true).EnableSorting(true).Allow
LabelFilter(true).AllowValueFilter(true).Rows(rows =>
{
 rows.Name("state").Add();
 rows.Name("eyeColor").Add();
})
.Columns(columns =>
{
 columns.Name("gender").Caption("Population").Add();
 columns.Name("isActive").Add();
})
.Values(values =>
{
 values.Name("balance").Caption("Balance").Add();
 values.Name("quantity").Caption("Quantity").Add();
})
.SortSettings(sortSettings =>
{

sortSettings.Name("company").Order(Syncfusion.EJ2.PivotView.Sorting.Descendi
ng).Add();
})
.FormatSettings(formatSettings =>
{
 formatSettings.Name("balance").Format("C").Add();
 formatSettings.Name("date").Format("dd/MM/yyyy-
hh:mm").Type("date").Add();
})
.DrilledMembers(drilledMembers =>
{

drilledMembers.Name("product").Items(ViewBag.drilledMembersProduct).Add();

drilledMembers.Name("gender").Items(ViewBag.drilledMembersGender).Add();
})
.FilterSettings(filterSettings =>
{

filterSettings.Name("date").Type(Syncfusion.EJ2.PivotView.FilterType.Date).C
ondition(Syncfusion.EJ2.PivotView.Operators.Between).Value1("new
Date('02/16/2000')").Value2("new Date('02/16/2002')").Add();

filterSettings.Name("age").Type(Syncfusion.EJ2.PivotView.FilterType.Number).
Condition(Syncfusion.EJ2.PivotView.Operators.Between).Value1("25").Value2("3
5").Add();

```

```

filterSettings.Name("eyeColor").Type(Syncfusion.EJ2.PivotView.FilterType.Exclude).Items(ViewBag.filterSettingsEyeColor).Add();
 }).ConditionalFormatSettings(format =>
 {

format.Measure("balance").Conditions(Syncfusion.EJ2.PivotView.Condition.Less Than).Value1(100000).Style(styles =>
 {

styles.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSize("12px");
 }
).Add();

format.Measure("quantity").Conditions(Syncfusion.EJ2.PivotView.Condition.Between).Value1(10).Value2(20).Style(style =>
 {

style.BackgroundColor("#f48fb1").Color("black").FontFamily("Tahoma").FontSize("12px");
 }
).Add();
 })
).ChartSettings(chartSettings =>
chartSettings.Value("Amount").EnableExport(true).EnableMultipleAxis(true).ChartSeries(
 chartSeries =>
chartSeries.Type(ChartSeriesType.Column).Animation(animation =>
 {
 animation.Enable(true);
 })
))
).DisplayOption(new PivotViewDisplayOption { View = View.Both
}).GroupingBarSettings(groupingBarSettings =>
{
 groupingBarSettings.AllowDragAndDrop(false);
}).GridSettings(gridSettings =>
{

gridSettings.ColumnWidth(140).ContextMenuItems(ViewBag.contextMenuItems);
}).GroupingBarSettings(groupingBarSettings => {
 groupingBarSettings.ShowFieldsPanel(true);
}).Toolbar(ViewBag.toolbar).Render();
<script>
 function saveReport(args) {
 var reports = [];
 var isSaved = false;
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reports = JSON.parse(localStorage.pivotviewReports);
 }
 if (args.report && args.reportName && args.reportName != '') {
 reports.map(function (item) {
 if (args.reportName === item.reportName) {
 item.report = args.report;
 isSaved = true;
 }
 });
 }
 }

```

```

 });
 if (!isSaved) {
 reports.push(args);
 }
 localStorage.pivotviewReports = JSON.stringify(reports);
}
}
function fetchReport(args) {
 var reportCollection = [];
 var reeportList = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 reeportList.push(item.reportName);
 });
 args.reportName = reeportList;
}
function loadReport(args) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 args.report = item.report;
 }
 });
 if (args.report) {
 pivotObj.dataSourceSettings =
JSON.parse(args.report).dataSourceSettings;
 }
}
function removeReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 for (var i = 0; i < reportCollection.length; i++) {
 if (reportCollection[i].reportName === args.reportName) {
 reportCollection.splice(i, 1);
 }
 }
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function renameReport(args) {
 var reportCollection = [];

```

```

 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 item.reportName = args.rename;
 }
 });
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 localStorage.pivotviewReports =
 JSON.stringify(reportCollection);
 }
 }
 function newReport() {
 var pivotObj =
 document.getElementById('pivotview').ej2_instances[0];
 pivotObj.setProperties({
 dataSourceSettings: {
 columns: [],
 rows: [],
 values: [],
 filters: []
 }
 }, false);
 }
 function beforeToolbarRender(args) {
 args.customToolbar.splice(12, 0, {
 prefixIcon: 'e-tool-expand e-icons', tooltipText:
 'Expand/Collapse',
 click: function (args) {
 var pivotTableObj =
 document.getElementById('pivotview').ej2_instances[0];
 pivotTableObj.dataSourceSettings.expandAll =
 !pivotTableObj.dataSourceSettings.expandAll;
 },
 });
 }
}
</script>

```

### ACCESSIBILITY.CS

```

public ActionResult Index()
{
 ViewBag.data = GetPivot_Data();
 ViewBag.drilledMembersProduct = new string[] { "Bike", "Car" };
 ViewBag.drilledMembersGender = new string[] { "male" };
 ViewBag.filterSettingsEyeColor = new string[] { "blue" };
 ViewBag.toolbar = new string[]
 {
 "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid",
 "Chart", "Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
 "NumberFormatting", "FieldList"
 };
 ViewBag.contextMenuItems = new string[]

```



```

 {
 "Aggregate", "CalculatedField", "Drillthrough", "Excel Export", "Pdf
Export", "Csv Export", "Expand", "Collapse", "Sort Ascending", "Sort
Descending"
 };
 return View();
}
public class Pivot_Data
{
 public string _id { get; set; }
 public int index { get; set; }
 public string guid { get; set; }
 public Boolean isActive { get; set; }
 public Double balance { get; set; }
 public int advance { get; set; }
 public int quantity { get; set; }
 public int age { get; set; }
 public string eyeColor { get; set; }
 public string name { get; set; }
 public string gender { get; set; }
 public string company { get; set; }
 public string email { get; set; }
 public string phone { get; set; }
 public string date { get; set; }
 public string product { get; set; }
 public string state { get; set; }
 public string pno { get; set; }
}
public List<Pivot_Data> GetPivot_Data()
{
 List<Pivot_Data> pivot_Data = new List<Pivot_Data>();
 pivot_Data.Add(new Pivot_Data{ _id = "5a940692c2d185d9fde50e5e", index =
0, guid = "810a1191-81bd-4c18-ac73-d16ad3fc80eb", isActive = false, balance
= 2430.87, advance = 7658, quantity = 11, age = 21, eyeColor = "blue", name
= "Skinner Ward", gender = "male", company = "GROK", email =
"skinnerward@grok.com", phone = "+1 (931) 600-3042", date = "Wed Feb 16 2000
15:01:01 GMT+0530 (India Standard Time)", product = "Flight", state = "New
Jersey", pno = "FEDD2340" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a940692c5752f1ed81bbb3d", index =
1, guid = "41c9986b-ccef-459e-a22d-5458bbdca9c7", isActive = true, balance =
3192.7, advance = 6124, quantity = 15, age = 27, eyeColor = "brown", name =
"Gwen Dixon", gender = "female", company = "ICOLOGY", email =
"gwendixon@icology.com", phone = "+1 (951) 589-2187", date = "Sun Feb 10
1991 20:28:59 GMT+0530 (India Standard Time)", product = "Jet", state =
"Vetaikan", pno = "ERTS4512" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a9406924c0e7f4c98a82ca7", index =
2, guid = "50d2bf16-9092-4202-84f6-e892721fe5a5", isActive = true, balance =
1663.84, advance = 7631, quantity = 14, age = 28, eyeColor = "green", name =
"Deena Gillespie", gender = "female", company = "OVERPLEX", email =
"deenagillespie@overplex.com", phone = "+1 (826) 588-3430", date = "Thu Mar
18 1993 17:07:48 GMT+0530 (India Standard Time)", product = "Car", state =
"New Jersey", pno = "ERTS4512" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a940692dd9db638eee09828", index =
3, guid = "b8bdc65e-4338-440f-a731-810186ce0b3a", isActive = true, balance =
1601.82, advance = 6519, quantity = 18, age = 33, eyeColor = "green", name =
"Susanne Peterson", gender = "female", company = "KROG", email =
"susannepeterson@krog.com", phone = "+1 (868) 499-3292", date = "Sat Feb 09

```

```

2002 04:28:45 GMT+0530 (India Standard Time)", product = "Jet", state =
"Vetaikan", pno = "CCOP1239" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a9406926f9971a87eae51af", index =
4, guid = "3f4c79ec-a227-4210-940f-162ca0c293de", isActive = false, balance
= 1855.77, advance = 7333, quantity = 20, age = 33, eyeColor = "green", name
= "Stokes Hicks", gender = "male", company = "SIGNITY", email =
"stokeshicks@signity.com", phone = "+1 (927) 585-2980", date = "Fri Mar 12
2004 11:08:06 GMT+0530 (India Standard Time)", product = "Van", state =
"Tamilnadu", pno = "MEWD9812" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a940692bcbbcdde08fcf7ec", index =
5, guid = "1d0ee387-14d4-403e-9a0c-3a8514a64281", isActive = true, balance =
1372.23, advance = 5668, quantity = 16, age = 39, eyeColor = "green", name =
"Sandoval Nicholson", gender = "male", company = "IDEALIS", email =
"sandovalnicholson@idealis.com", phone = "+1 (951) 438-3539", date = "Sat
Aug 30 1975 22:02:15 GMT+0530 (India Standard Time)", product = "Bike",
state = "Tamilnadu", pno = "CCOP1239" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a940692ff31a6e1cdd10487", index =
6, guid = "58417d45-f279-4e21-ba61-16943d0f11c1", isActive = false, balance
= 2008.28, advance = 7107, quantity = 14, age = 20, eyeColor = "brown", name
= "Blake Thornton", gender = "male", company = "IMMUNICS", email =
"blakethornton@immunics.com", phone = "+1 (852) 462-3571", date = "Mon Oct
03 2005 05:16:53 GMT+0530 (India Standard Time)", product = "Jet", state =
"New Jercy", pno = "CCOP1239" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a9406928f2f2598c7ac7809", index =
7, guid = "d16299e3-e243-4e57-90fb-52446c4c0275", isActive = false, balance
= 2052.58, advance = 7431, quantity = 20, age = 22, eyeColor = "blue", name
= "Dillard Sharpe", gender = "male", company = "INEAR", email =
"dillardsharpe@inear.com", phone = "+1 (963) 473-2308", date = "Thu May 25
1978 04:57:00 GMT+0530 (India Standard Time)", product = "Jet", state =
"Rajkot", pno = "ERTS4512" });
 pivot_Data.Add(new Pivot_Data{ _id = "5a940692195f82585af58362", index =
8, guid = "a8662b61-9d66-4b99-8a47-c0375ffff4ce", isActive = true, balance =
2784.64, advance = 7948, quantity = 18, age = 22, eyeColor = "blue", name =
"Davidson Brewer", gender = "male", company = "PROXSOF", email =
"davidsonbrewer@proxsoft.com", phone = "+1 (958) 592-3948", date = "Wed Jul
14 1982 19:49:29 GMT+0530 (India Standard Time)", product = "Van", state =
"Vetaikan", pno = "FEDD2340" });
 return pivot_Data;
}

```

See also

- [Accessibility in Syncfusion Pivot control](#)

## Migration from Essential JS 1

This article describes the API migration process of Pivot Grid component from Essential JS 1 to Essential JS 2.

### Data Binding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Data source | **property:**

```
dataSource

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Data((IEnumerable<object>)ViewBag.Data))| property:
dataSourceSettings

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource
e => dataSource.DataSource((IEnumerable<object>)ViewBag.Data)).Render()|
```

| Rows | **property:**

```
rows

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("Country").FieldCaption("Country").Add(); })))|
property: rows

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.Rows(rows =>{ rows.Name("Country").Caption("Country").Add(); })).Render()|
```

| Columns | **property:**

```
columns

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Columns(columns => {
columns.FieldName("Country").FieldCaption("Country").Add(); })))| property:
columns

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.Columns(columns =>{ columns.Name("Country").Caption("Country").Add();
})).Render()|
```

| Values | **property:**

```
values

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Values(values => { values.FieldName("balance").FieldCaption("Balance($)").Add();
})))| property:
values

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.Values(values =>{ values.Name("balance").Caption("Balance($)").Add();
})).Render()|
```

| Filters | **property:**

```
filters

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Filters(filters => { filters.FieldName("Country").FieldCaption("Country").Add();
})))| property:
filters

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.Filters(filters =>{ filters.Name("Country").Caption("Country").Add(); })).Render()|
```

| Value axis position | Not Applicable | **property:**

```
valueAxis

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.ValueAxis("row")).Render()|
```

## Aggregation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Summary Type | **property:**

```
summaryType

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Values(values => {
values.FieldName("balance").FieldCaption("Balance($)").SummaryType("Count").Add();
})))| property: type

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource
```

```
=> dataSource.Values(values =>{ values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold
Amount").Type(SummaryType.Count).Add();}).Render() |
```

### Number Format

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Format settings | **property:**

```
format

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Values(values => {
values.FieldName("balance").FieldCaption("Balance($)").Format("currency").Add(); })| property:
formatSettings

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.FormatSettings(formatsettings => {
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignific
antDigits(1).UseGrouping(true).Add(); formatsettings.Name("Year").Format("dd/MM/yyyy-
hh:mm").Type("date").Add(); })).Render() |
```

### Summary Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide grand totals | **property:**

```
enableGrandTotal

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.EnableGrandTotal(false)) | property:
showGrandTotals

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource
=> dataSource.ShowGrandTotals(false)).Render() |
```

| Show/hide row grand totals | **property:**

```
enableRowGrandTotal

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource
=> dataSource.EnableRowGrandTotal(false)) | property:
showRowGrandTotals

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource
=> dataSource.ShowRowGrandTotals(false)).Render() |
```

| Show/hide column grand totals | **property:**

```
enableColumnGrandTotal

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource
=> dataSource.EnableColumnGrandTotal(false)) | property:
showColumnGrandTotals

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource
=> dataSource.ShowColumnGrandTotals(false)).Render() |
```

| Show/hide sub-totals | Not Applicable | **property:**

```
showSubTotals

@Html.EJS().PivotView("PivotView").DataSource(dataSource =>
dataSource.ShowSubTotals(false)).Render() |
```

| Show/hide row sub-totals | Not Applicable | **property:**

```
showRowSubTotals

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource
=> dataSource.ShowRowSubTotals(false)).Render() |
```

| Show/hide column sub-totals | Not Applicable | **property:**

showColumnSubTotals<br/><br/>@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.ShowColumnSubTotals(false)).Render() |

| Show/hide sub-totals for specific field | **property:**

showSubTotal<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource => dataSource.Rows(rows => { rows.FieldName("Country").ShowSubTotal(false).Add(); } ) |

**property:**

showSubTotals<br/><br/>@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.Rows(rows => { rows.Name("Country").ShowSubTotals(false).Add(); } )).Render() |

### Drill operation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Expand All | **property:**

enableCollapseByDefault<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCollapseByDefault(true) | **property:**

expandAll<br/><br/>@Html.EJS().PivotView("PivotView").ExpandAll(true).Render() |

| Drill Up/Down | **property:**

collapsedMembers<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEvents => clientSideEvents.Load("onLoad"))<br/><br/>function onLoad(args)

{<br/>args.model.collapsedMembers = { Country: ["Canada", "France"] }<br/>} | **property:**

drilledMembers<br/><br/>@Html.EJS().PivotView("PivotView").DrilledMembers(drilledmembers => { drilledmembers.Name("Country").Items(ViewBag.drilledMembers).Add(); } ).Render() |

### Field List

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide field list pop-up button on pivot grid | Not Applicable | **property:**

showFieldList<br/><br/>@Html.EJS().PivotView("PivotView").ShowFieldList(true).Render() |

| Defer update | **property:**

enableDeferUpdate<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableDeferUpdate(true) | Not Applicable |

| Control initialization | **component:**

PivotSchemaDesigner<br/><br/>@Html.EJ().Pivot().PivotSchemaDesigner("PivotSchemaDesigner") | **component:**

PivotFieldListComponent<br/><br/>@Html.EJS().PivotFieldList("PivotFieldList").Render() |

| Render mode | Not Applicable | **property:**

renderMode<br/><br/>@Html.EJS().PivotFieldList("PivotFieldList").RenderMode("Fixed").Render() |

| Show/hide calculated field button | Not Applicable | **property:**

allowCalculatedField<br/><br/>@Html.EJS().PivotFieldList("PivotFieldList").AllowCalculatedField(true).Render() |

| Show/hide value group button | Not Applicable | **property:**

```
showValuesButton

@Html.EJS().PivotFieldList("PivotFieldList").ShowValuesButton(true)
.Render()|
```

### Grouping Bar

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide Grouping bar | **property:**

```
enableGroupingBar

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableGroupingBar(true)
```

| **property:**

```
showGroupingBar

@Html.EJS().PivotView("PivotView").ShowGroupingBar(true).Render()|
```

| Grouping Bar Settings | Not Applicable | **property:**

```
groupingBarSettings

@Html.EJS().PivotView("PivotView").GroupingBarSettings(new
PivotViewGroupingBarSettings { ShowFilterIcon =false ShowSortIcon=true
ShowRemovelcon=false}).Render()|
```

| Show/hide value group button | Not Applicable | **property:**

```
showValuesButton

@Html.EJS().PivotView("PivotView").ShowValuesButton(true).Render()|
```

### Filtering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Filter settings | **property:**

```
filterItems

@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource =>
dataSource.Rows(rows => { rows.FieldName("Country").FieldCaption("Country").Add();
})).ClientSideEvents(clientSideEvents => clientSideEvents.Load("onLoad"))

function
onLoad(args) {
args.model.dataSource.rows[0].filterItems = {
filterType:
ej.PivotAnalysis.FilterType.Exclude,
 values: ["Canada", "France"] };
}| property:
filterSettings

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource =>
dataSource.FilterSettings(filtersettings =>{
filtersettings.Name("Country").Type("Exclude").Items(ViewBag.filterMembers).Add();})).Render()|
```

### Maximum node limit in member editor

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Max node limit in member editor | **property:**

```
maxNodeLimitInMemberEditor

@Html.EJ().Pivot().PivotGrid("PivotGrid").MaxNodeLimitInMemberEditor(100)| property:
```

```
maxNodeLimitInMemberEditor

@Html.EJS().PivotView("PivotView").MaxNodeLimitInMemberEditor(100).Render()|
```

## No Data Items

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Show/hide "no data" items | Not Applicable | **property:**

```
showNoDataItems

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.Rows(rows =>{ rows.Name("Country").showNoDataItems(true).Add(); })).Render();
```

## Excel-like filtering

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Label filtering | **property:**

```
enableAdvancedFilter

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableAdvancedFilter(true).DataSource(dataSource => dataSource.Rows(rows => { rows.FieldName("Country").FieldCaption("Country").Add(); })).ClientSideEvents(clientSideEvents => clientSideEvents.Load("onLoad"))

function onLoad(args) {
args.model.dataSource.rows[0].advancedFilter = [{
 labelFilterOperator: ej.olap.LabelFilterOptions.EndsWith,
values: ["s"]}]
}| property: allowLabelFilter

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.AllowLabelFilter(true).FilterSettings(filtersettings =>{ filtersettings.Name("Country").Type(FilterType.Label).Condition(Operators.GreaterThan).Value 1('United Kingdom').Add();})).Render();
```

| Value filtering | **property:**

```
enableAdvancedFilter

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableAdvancedFilter(true).DataSource(dataSource => dataSource.Rows(rows => { rows.FieldName("Country").FieldCaption("Country").Add(); })).ClientSideEvents(clientSideEvents => clientSideEvents.Load("onLoad"))

function onLoad(args) {
args.model.dataSource.rows[0].advancedFilter = [{
 measure: "balance"
valueFilterOperator: ej.olap.ValueFilterOptions.EndsWith,
values: ["002"]}]
}| property: allowValueFilter

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.AllowValueFilter(true).FilterSettings(filtersettings =>{ filtersettings.Name("Country").Measure("quantity").Type(FilterType.Value).Condition(Operators.Between).Value1("3250")..Value2("5000").Add();})).Render();
```

## Drill Through

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Show/hide drill though feature | **property:**

```
enableDrillThrough

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableDrillThrough(true)| property: allowDrillThrough

@Html.EJS().PivotView("PivotView").AllowDrillThrough(true).Render();
```



| Event Triggers when cell clicked in pivot grid control | **event:**  
 drillThrough<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEvents => clientSideEvents.DrillThrough("onDrillThrough"))<br/><br/>function onDrillThrough() {  
 }| **event:**  
 drillThrough<br/><br/>@Html.EJS().PivotView("PivotView").DrillThrough("onDrillThrough").Render()  
 <br/><br/>function onDrillThrough() { }|

### Cell Editing

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Edit settings | Not Applicable | **property:**  
 editSettings<br/><br/>@Html.EJS().PivotView("PivotView").EditSettings(new  
 PivotViewCellEditSettings{}).Render()|

| Show/hide cell editing feature | **property:**  
 enableCellEditing<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCellEditing(true) | **property:** allowEditing<br/><br/>@Html.EJS().PivotView("PivotView").EditSettings(new  
 PivotViewCellEditSettings{AllowAdding=true;AllowDeleting=true;AllowEditing=true;Mode=EditMode.Dialog}).Render()|

### Hyperlink

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Hyperlink settings | **property:**  
 hyperlinkSettings<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").HyperlinkSettings() | **property:** hyperlinkSettings<br/><br/>@Html.EJS().PivotView("PivotView").HyperlinkSettings().Render()|

| Show/hide hyperlink to all cells | Not Applicable | **property:**  
 showHyperlink<br/><br/>@Html.EJS().PivotView("PivotView").HyperlinkSettings( new  
 PivotViewHyperLinkSettings { showHyperlink =true }).Render()|

| Show/hide hyperlink to row headers | **property:**  
 enableRowHeaderHyperlink<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").HyperlinkSettings  
 (hyperLink => hyperLink.EnableRowHeaderHyperlink(true)) | **property:**  
 showRowHeaderHyperlink<br/><br/>@Html.EJS().PivotView("PivotView").HyperlinkSettings( new  
 PivotViewHyperLinkSettings { showRowHeaderHyperlink =true }).Render()|

| Show/hide hyperlink to column headers | **property:**  
 enableColumnHeaderHyperlink<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").HyperlinkSettings  
 (hyperLink => hyperLink.EnableColumnHeaderHyperlink(true)) | **property:**  
 showColumnHeaderHyperlink<br/><br/>@Html.EJS().PivotView("PivotView").HyperlinkSettings(  
 new PivotViewHyperLinkSettings { showColumnHeaderHyperlink =true }).Render()|

| Show/hide hyperlink to value cells | **property:**  
 enableValueCellHyperlink<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").HyperlinkSettings(h  
 yperLink => hyperLink.EnableValueCellHyperlink(true)) | **property:**



```
showValueCellHyperlink

@Html.EJS().PivotView("PivotView").HyperlinkSettings(new
PivotViewHyperLinkSettings { showValueCellHyperlink =true }).Render()|
```

| Show/hide hyperlink to summary cells| **property:**

```
enableSummaryCellHyperlink

@Html.EJ().Pivot().PivotGrid("PivotGrid").HyperlinkSetting
s(hypLink => hypLink.EnableSummaryCellHyperlink(true))| property:
showSummaryCellHyperlink

@Html.EJS().PivotView("PivotView").HyperlinkSettings(
new PivotViewHyperLinkSettings { showSummaryCellHyperlink =true }).Render()|
```

| Show/hide hyperlink using specific conditions| Not Applicable| **property:**

```
conditionalSettings

@Html.EJS().PivotView("PivotView").HyperlinkSettings(hyperlinkset
tings => hyperlinksettings..ConditionalSettings(format
=>{format.Conditions(Condition.Between).Measure("Units
Sold").Value1(150).Value2(200).Add();}).Render()|
```

| Show/hide hyperlink for row or column| Not Applicable| **property:**

```
headerText

@Html.EJS().PivotView("PivotView").HyperlinkSettings(new
PivotViewHyperLinkSettings { headerText = 'FY 2015.Q1.Units Sold' }).Render()|
```

| Event Triggers when row headers clicked in pivot grid control| **event:**

```
rowHeaderHyperlinkClick

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(cl
ientSideEvents => clientSideEvents.RowHeaderHyperlinkClick("onClick"))

function
onClick() { }| event:
hyperlinkCellClick

@Html.EJS().PivotView("PivotView").HyperlinkCellClick("onClick").Re
nder()

function onClick() { }|
```

| Event Triggers when column headers clicked in pivot grid control| **event:**

```
columnHeaderHyperlinkClick

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvent
s(clientSideEvents =>
clientSideEvents.ColumnHeaderHyperlinkClick("onClick"))

function onClick() { }| event:
hyperlinkCellClick

@Html.EJS().PivotView("PivotView").HyperlinkCellClick("onClick").Re
nder()

function onClick() { }|
```

| Event Triggers when value cells clicked in pivot grid control| **event:**

```
valueCellHyperlinkClick

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clie
ntSideEvents => clientSideEvents.ValueCellHyperlinkClick("onClick"))

function
onClick() { }| event:
hyperlinkCellClick

@Html.EJS().PivotView("PivotView").HyperlinkCellClick("onClick").Re
nder()

function onClick() { }|
```

| Event Triggers when summary cells clicked in pivot grid control| **event:**

```
summaryCellHyperlinkClick

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(
clientSideEvents => clientSideEvents.SummaryCellHyperlinkClick("onClick"))

function
onClick() { }| event:
hyperlinkCellClick

@Html.EJS().PivotView("PivotView").HyperlinkCellClick("onClick").Re
nder()

function onClick() { }|
```

[Defer Layout Update](#)

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide defer layout update | **property:**

enableDeferUpdate<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableDeferUpdate(true)

| **property:**

allowDeferLayoutUpdate<br/><br/>@Html.EJS().PivotView("PivotView").AllowDeferLayoutUpdate(true).Render() |

### Sorting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable/disable sorting | Not Applicable | **property:**

enableSorting<br/><br/>@Html.EJS().PivotView("PivotView").EnableSorting(true).Render() |

| Sort settings | **property:**

sortOrder<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").DataSource(dataSource => dataSource.Rows(rows => { rows.FieldName("Country").FieldCaption("Country").SortOrder(SortOrder.Descending).Add(); }))) | **property:**

sortSettings<br/><br/>@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.SortSettings(sortsettings => { sortsettings.Name("company").Order("Descending").Add(); }))).Render() |

### Value Sorting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable/disable value sorting | Not Applicable | **property:**

enableSorting<br/><br/>@Html.EJS().PivotView("PivotView").EnableValueSorting(true).Render() |

| Value sort settings | **property:**

valueSortSettings<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid").ValueSortSettings(valuesort settings=>valuesortsettings.HeaderText("Bike##Quantity").HeaderDelimiters("##").SortOrder(SortOrder.Descending)) | **property:**

valueSortSettings<br/><br/>@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.ValueSortSettings(new PivotViewValueSortSettings { HeaderText = "FY 2015##Sold Amount", HeaderDelimiter = "##", SortOrder = Sorting.Descending })).Render() |

### Calculated Field

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Show/hide calculated field | Not Applicable | **property:**

allowCalculatedField<br/><br/>@Html.EJS().PivotView("PivotView").AllowCalculatedField(true).Render() |

| Calculated field settings | **property:**

values<br/><br/>@Html.EJ().Pivot().PivotGrid("PivotGrid1").DataSource(dataSource => dataSource.Values(values => { values.FieldName("Amount").Add();

```
values.FieldName("Price").FieldCaption("Price").IsCalculatedField(true).Formula("Amount * 15").Add(); })|property:
calculatedFieldSettings

@Html.EJS().PivotView("PivotView").DataSourceSettings(dataSource => dataSource.CalculatedFieldSettings(calculatedfieldsettings => {
calculatedfieldsettings.Name("Total").Formula(totalPrice).Add(); })).Render()

@{var
stock = "\" + "Sum(In_Stock)" + "\"";}
@{var sold = "\" + "Sum(Sold)" + "\"";}
@{ var
totalPrice = stock + "+" + sold;}|
```

### Paging

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Paging|**property:**

```
enablePaging

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnablePaging(true)|Not
Applicable|
```

|Virtual scrolling|**property:**

```
enableVirtualScrolling

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableVirtualScrolling(t
rue)|property:
enableVirtualization

@Html.EJS().PivotView("PivotView").EnableVirtualization(true).Re
nder()|
```

### Conditional Formatting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Show/hide conditional formatting|**property:**

```
enableConditionalFormatting

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCondition
alFormatting(true)|property:
allowConditionalFormatting

@Html.EJS().PivotView("PivotView").AllowConditionalForm
atting(true).Render()|
```

|Conditional formatting settings|**property:**

```
conditionalFormatSettings

@Html.EJ().Pivot().PivotGrid("PivotGrid").ClientSideEvents(cli
entSideEvents => clientSideEvents.Load("onLoad"))

function onLoad(args)
{
args.model.dataSource.conditionalFormatSettings = [{
name: "Format2",
 style: {

"color": "#000000",
 "backgroundcolor": "#0000FF",
 "bordercolor":
"#000000",
 "borderstyle": "Dashed",
 "borderwidth": "5",
 "fontsize": "12",

"fontstyle": "Algerian" },
condition: ej.PivotGrid.ConditionalOptions.LessThan,
 value:
"200",
 measures: "Amount,Quantity" }]
}|property:
conditionalFormatSettings

@Html.EJS().PivotView("PivotView").DataSourceSettings(dat
aSource => dataSource..ConditionalFormatSettings(format =>{
format.Conditions(Condition.LessThan).Measure("In Stock").Value1(5000).Style(style => {
style.BackgroundColor("#80cbc4").Color("black").FontFamily("Tahoma").FontSize("12px");
}).Add(); format.Conditions(Condition.Between).Measure("Units
Sold").Value1(3400).Value2(40000).Style(style => {
```

```
style.BackgroundColor("#f48fb1").Color("black").FontFamily("Tahoma").FontSize("12px");
}).Add(); })).Render();
```

### Exporting

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Excel Export | Not Applicable | **property:**

```
allowExcelExport

@Html.EJS().PivotView("PivotView").AllowExcelExport(true).Render()
|
```

| Pdf Export | Not Applicable | **property:**

```
allowPdfExport

@Html.EJS().PivotView("PivotView").AllowPdfExport(true).Render() |
```

### Grid Customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Set width for pivot grid | Not Applicable | **property:**

```
width

@Html.EJS().PivotView("PivotView").Width(800).Render() |
```

| Set height for pivot grid | Not Applicable | **property:**

```
height

@Html.EJS().PivotView("PivotView").Height(400).Render() |
```

| Set row height for pivot grid | Not Applicable | **property:**

```
rowHeight

@Html.EJS().PivotView("PivotView").GridSettings(new
PivotViewGridSettings { RowHeight = 60 }).Render() |
```

| Set column width for pivot grid | Not Applicable | **property:**

```
columnWidth

@Html.EJS().PivotView("PivotView").GridSettings(new
PivotViewGridSettings { ColumnWidth = 120 }).Render() |
```

| Drag and drop column headers in pivot grid | Not Applicable | **property:**

```
allowReordering

@Html.EJS().PivotView("PivotView").GridSettings(new
PivotViewGridSettings { AllowReordering = true }).Render() |
```

| Resizing the column headers in pivot grid | **property:**

```
enableColumnResizing

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableColumnResizing
(true) | property: allowResizing

@Html.EJS().PivotView("PivotView").GridSettings(new
PivotViewGridSettings { AllowResizing = true }).Render() |
```

| Wrap the cell content in pivot grid | Not Applicable | **property:**

```
allowTextWrap

@Html.EJS().PivotView("PivotView").GridSettings(new
PivotViewGridSettings { AllowTextWrap = true }).Render() |
```

| Display cell border in pivot grid | Not Applicable | **property:**

```
gridLines

@Html.EJS().PivotView("PivotView").GridSettings(new PivotViewGridSettings
{ GridLines = "Vertical" }).Render() |
```

| Cell selection | **property:**

```
enableCellSelection

@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCellSelection(true)
| property: allowSelection

@Html.EJS().PivotView("PivotView").GridSettings(new
```

```
PivotViewGridSettings { AllowSelection = true }).Load("onLoad").Render()
onLoad() {
 pivotObj = document.getElementById('PivotView').ej2_instances[0];
 pivotObj.gridSettings.selectionSettings = {
 cellSelectionMode: 'Box',
 type: 'Multiple',
 mode: 'Cell' };
```

| Display overflow cell content in pivot grid | Not Applicable | **property:**  
clipMode  
@Html.EJS().PivotView("PivotView").GridSettings(new PivotViewGridSettings { ClipMode = "Clip" }).Render();

| Cell Editing | **property:**  
enableCellEditing  
@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCellEditing(true) | Not Applicable

| Cell double click | **property:**  
enableCellDoubleClick  
@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCellDoubleClick(true) | Not Applicable

| Cell context | **property:**  
enableCellContext  
@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableCellContext(true) | Not Applicable

#### Accessibility

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Localization | **property:** locale  
@Html.EJ().Pivot().PivotGrid("PivotGrid").Locale("es-ES") | **property:** locale  
@Html.EJS().PivotView("PivotView").Locale("es-ES").Render();

| Right to left | **property:**  
enableRTL  
@Html.EJ().Pivot().PivotGrid("PivotGrid").EnableRtl(true) | **property:**  
enableRtl  
@Html.EJS().PivotView("PivotView").EnableRTL(true).Render();

#### Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Adding custom class to wrapper element | **property:**  
cssClass  
@Html.EJ().Pivot().PivotGrid("PivotGrid").CssClass("custom-class") | **property:**  
cssClass  
@Html.EJS().PivotView("PivotView").CssClass("custom-class").Render();

| Keeping the model values in cookies | Not Applicable | **property:**  
enablePersistence  
@Html.EJS().PivotView("PivotView").EnablePersistence(true).Render();

| Event triggers when control initializing | **event:**  
load  
@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEvents => clientSideEvents.Load("onLoad"))  
function onLoad() { } | **event:**  
load  
@Html.EJS().PivotView("PivotView").Load("onLoad").Render()  
function onLoad() { }

| Event Triggers before the pivot engine starts | **event:**

```
beforePivotEnginePopulate

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(
clientSideEvents =>
clientSideEvents.BeforePivotEnginePopulate("onBeforePivotEnginePopulate"))

function
onBeforePivotEnginePopulate() { }| event:
enginePopulating

@Html.EJS().PivotView("PivotView").EnginePopulating("onEnginePop
ulating").Render()

function onEnginePopulating() { }|
```

| Event Triggers after the pivot engine populated | **event:**

```
afterPivotEnginePopulate

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(c
lientSideEvents =>
clientSideEvents.AfterPivotEnginePopulate("onAfterPivotEnginePopulate"))

function
onAfterPivotEnginePopulate() { }| event:
enginePopulated

@Html.EJS().PivotView("PivotView").EnginePopulated("onEnginePop
ulated").Render()

function onEnginePopulated() { }|
```

| Event Triggers after the control populated with data source | **event:**

```
renderSuccess

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEv
ents => clientSideEvents.RenderSuccess("onRenderSuccess"))

function
onRenderSuccess() { }| event:
dataBound

@Html.EJS().PivotView("PivotView").DataBound("onDataBound").Render()

function onDataBound() { }|
```

| Event Triggers after the control created | Not Applicable | **event:**

```
created

@Html.EJS().PivotView("PivotView").Created("onCreated").Render()

function onCreated() { }|
```

| Event Triggers when destroy the control | Not Applicable | **event:**

```
destroyed

@Html.EJS().PivotView("PivotView").Destroyed("onDestroyed").Render()

/>
function onDestroyed() { }|
```

| Event Triggers the cell clicked in pivot grid control | **event:**

```
cellClick

@Html.EJ().Pivot().PivotGrid("PivotGrid1").ClientSideEvents(clientSideEvents
=> clientSideEvents.CellClick("onCellClick"))

function onCellClick() { }| event:
cellClick

@Html.EJS().PivotView("PivotView").CellClick("onCellClick").Render()

>function onCellClick() { }|
```

## How To

```
<!-- markdownlint-disable MD009 -->
```

### Switching to older themes style

From Volume 1, 2020 onwards Syncfusion has revised the theming and layout of the Pivot Table. So, to inherit the older theme style and layout do the necessary changes in CSS and pivot table height.

### CSS Selectors

In current theme, the cells can be differentiated by their background colors. To avoid it, you need to override its background colors via simple CSS coding within your application. The below CSS selectors allow to achieve the same for material, fabric, bootstrap and bootstrap v4 themes.

```
`html
```

```

<!-- Codes here... -->
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#fff !important;
}
</style>
`

```

Meanwhile for high contrast theme, we need to set the following CSS.

```

`html
<!-- Codes here... -->
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#000 !important;
}
</style>
`

```

### *Adjusting Row Height*

In current theme, to make the component compact we have reduced the height of each pivot table rows. But user can reset the height of the pivot table using the [RowHeight](#) property in [PivotViewGridSettings](#). In older theme, the property was set to 36 pixels for desktop layout and 48 pixels for mobile layout. So reset the [RowHeight](#) accordingly to visualize the older theme style.

In the below code sample, we replicate the older theme style.

### **CSHTML**

```
@using Syncfusion.EJ2.PivotView
```



```

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(new PivotViewGridSettings { RowHeight = 36 }).Render()
<style>
.e-pivotview .e-rowsheader,
.e-pivotview .e-columnsheader,
.e-pivotview .e-gtot,
.e-pivotview .e-content,
.e-pivotview .e-gridheader,
.e-pivotview .e-headercell {
background-color:#fff !important;
}
</style>

```

### SWITCHTHEME.CS

```

public ActionResult Index()
{
var data = GetPivotData();
ViewBag.DataSource = data;
return View();
}

```



|                    | FY 2015     |                    | FY 2016     |                    | FY 2017    |
|--------------------|-------------|--------------------|-------------|--------------------|------------|
|                    | Units Sold  | Sold Amount        | Units Sold  | Sold Amount        | Units Sold |
| France             | 450         | \$714,955          | 526         | \$1,542,104        |            |
| Germany            | 440         | \$563,515          | 496         | \$1,772,104        |            |
| United States      | 546         | \$754,515          | 636         | \$2,263,104        |            |
| <b>Grand Total</b> | <b>1436</b> | <b>\$2,032,585</b> | <b>1658</b> | <b>\$5,577,312</b> |            |

### Customize number, date, and time values

You can format the number, date, and time values for each field using [FormatSettings](#) option under [PivotViewDataSourceSettings](#). It can be configured through code behind, during initial rendering.

#### Number formatting

For numbers, the formatting settings required to apply through code behind are:

- **name**: It allows to set the field name.
- **format**: It allows to set the format of the respective field.

**Note:** Also, you can customize the applied number format by setting the [NumberFormatOptions](#) options in [FormatSettings](#) itself.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(dataSource =>
 dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource)
 .ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Amount").Format("C2").MaximumSignificantDigits(3).MinimumSignificantDigits(1).UseGrouping(false).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).Render()
```

**NUMBER.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

*Date and time formatting*

**Note:** This property is applicable only for relational data source.

For date and time, the formatting settings required to apply through code behind are:

- **name:** It allows to set the field name.
- **format:** It allows to set the format of the respective field.
- **type:** It allows to set the type of format to be used for the respective field.

**Note:** Also, you can customize the applied date format by setting [DateFormatOptions](#) options in [FormatSettings](#) itself.

**CSHTML**

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").DataSourceSettings(
 dataSource =>
 {
 dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource)
 .ExpandAll(false).EnableSorting(true)
 .FormatSettings(formatsettings =>
 {
 formatsettings.Name("Year").Format("dd/MM/yyyy-hh:mm").Type("date").Add();
 })
 .Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add(); columns.Name("Quarter").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
 })
.Render()
```

**DATE.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

**Limitations of date formatting**

As per Firefox and Edge browsers standards, most of the date and time formats used in data source aren't supported. For example: Apr-2000, Apr-01-2000, 01-03-2000, 2000-Apr-01 etc... are not supported. Meanwhile [ISO formats](#) will be supported across all browsers.

### Customize the icons for pivot table

You can customize the pivot button icons in the pivot table by overriding the class **.pivot-button** with a custom property content as mentioned below.

```
`html
<style>
PivotView_PivotFieldList .e-icons.e-toggle-field-list::before {
content: '\e337';
}
</style>
`
```

In the below sample, pivot table is rendered with a customized pivot button icons.

### CSHTML

```
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).ShowGroupingBar(true).Render()
<style>
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-dropdown-icon::before,
#PivotView .e-pivot-button .e-icons.e-dropdown-icon::before {
content: "\ec25";
}
#PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-pv-filtered::before,
#PivotView .e-pivot-button .e-icons.e-pv-filtered::before {
content: "\e611";
}
```

```

 #PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-pv-
filter::before,
 #PivotView .e-pivot-button .e-icons.e-pv-filter::before {
 content: '\e611';
 }
 #PivotView_PivotFieldList .e-icons.e-toggle-field-list::before {
 content: '\e342';
 }
 #PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-
sort::before,
 #PivotView .e-pivot-button .e-icons.e-sort::before {
 content: '\e523';
 }
 #PivotView_PivotFieldList_Wrapper .e-pivot-button .e-icons.e-
remove::before,
 #PivotView .e-pivot-button .e-icons.e-remove::before {
 content: '\e94a';
 }
 #PivotView_PivotFieldList_Wrapper .e-icons.e-drag::before,
 #PivotView .e-pivot-button .e-icons.e-drag::before {
 content: '\e571';
 }
 #PivotView .e-icons.e-expand::before {
 content: '\e22c';
 }
 #PivotView .e-icons.e-collapse::before {
 content: '\e22b';
 }
 }
</style>

```

### ICONCUSTOMIZE.CS

```

public IActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Customize empty value cells

You can show the custom string in the empty value cells using the [EmptyCellsTextContent](#) string type property in [PivotViewDataSourceSettings](#) object of the pivot table. It can be configured through code behind during initial rendering.

### CSHTML

```

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{

```

```
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).EmptyCellsTextContent("*").Render()
```

### **CUSTOMTEXT.CS**

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Configure data grid options on editing mode

You can access the data grid options such as sort, group, filter on editing mode using the [BeginDrillThrough](#) event in the pivot table. The event occurs in every value cell on double click and provides the data grid information before display the drill through grid pop-up.

### **CSHTML**

```
@Html.EJS().PivotView("PivotView").Width("100%").Height("300").ShowTooltip(false).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })
).EditSettings(pivotViewCellEditSettings=>pivotViewCellEditSettings.AllowA
```

```

dding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.Pivot
View.EditMode.Normal)).BeginDrillThrough("beginDrillThrough").Render()
<script>
 function beginDrillThrough(args) {
 if (args.gridObj) {
 var gridObj = args.gridObj;
 gridObj.allowGrouping = true;
 gridObj.allowSorting = true;
 gridObj.allowFiltering = true;
 gridObj.filterSettings = { type: 'CheckBox' };
 }
 }
</script>

```

### CUSTOMEDITING.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

Refresh the field list while change the data source

You can refresh pivot table and field list with new data source dynamically.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotGrid").Height("300").DataSource(dataSource =>
dataSource.Data((IEnumerable<object>)ViewBag.Data).ExpandAll(false).EnableSo
rting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").UseGrouping(true).Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).ShowFieldList(true).Render()
<script>
 var pivotGridObj;
 document.getElementById('Refresh').addEventListener('click', () => {
 pivotGridObj =
document.getElementById('PivotGrid').ej2_instances[0];
 pivotGridObj.engineModule.fieldList = {};
 pivotGridObj.dataSourceSettings.dataSource = [
 { 'Sold': 31, 'Amount': 52824, 'Country': 'France', 'Year': 'FY
2015' },

```

```

2015' },
 { 'Sold': 90, 'Amount': 153360, 'Country': 'France', 'Year': 'FY
2015' },
 { 'Sold': 25, 'Amount': 42600, 'Country': 'France', 'Year': 'FY
2015' },
 { 'Sold': 27, 'Amount': 46008, 'Country': 'France', 'Year': 'FY
2016' }]];
});
</script>

```

## REFRESH.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.data = data;
 return View();
}

```

## Customizing loading indicator

You can customize the appearance of the loading indicator in the pivot table by using the [SpinnerTemplate](#) property. This property accepts an HTML string which can be used for appearance customization.

**Note:** You can also disable the loading indicator by setting [SpinnerTemplate](#) to empty string.

## CSHTML

```

@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false).EnableSorting(true).AllowLabelFilter(true).AllowValueFilter(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}).ShowFieldList(true).SpinnerTemplate("<i class='fa fa-cog fa-spin fa-3x fa-fw'></i>").Render()

```

## FIELDLIST.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Changing the Pivot Table component's minimum height

The `minHeight` property allows you to change the minimum height for the pivot table control. For the pivot table control, the default minimum height is **300px**.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("200").Width(650).DataSourceSettings(
dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
.Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DataBound("ondataBound").Render()
<script>
 function ondataBound(args) {
 var pivotTableObj =
document.getElementById('PivotView').ej2_instances[0];
 pivotTableObj.minHeight = 200;
 }
</script>
```

#### MINHEIGHT.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

### Render chart control based on cell selection

The cell selection support is enabled using the [AllowSelection](#) property and its type and mode are configured using the [PivotViewSelectionSettings](#) property. The [cellSelected](#) event gets fired on every selection operation performed in the pivot table. This event returns the selected cell informations, like row header name, column header name, measure name, and value. Based on this information, the [chart](#) control will be plotted.



**CSHTML**

```

@using Syncfusion.EJ2.PivotView;
<div class="control-section" style="overflow:auto">
<div class="content-wrapper">

@Html.EJS().PivotView("pivotview").Width("100%").ShowTooltip(false).Height("
300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>
)ViewBag.Data).ExpandAll(true).EnableSorting(true)
 .Rows(rows =>
 {
 rows.Name("Country").Add();
 rows.Name("Products").Add();
 })
 .Columns(columns =>
 {
 columns.Name("Year").Add();
 })
 .Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 })).GridSettings(new PivotViewGridSettings { ColumnWidth = 140,
AllowSelection = true
}).DataBound("onDataBound").Load("onLoad").CellSelected("onCellSelected").Re
nder()
</div>

<div id="Chart" style="height: 450px;"></div>
</div>
<script>
 var onInit = true;
 var measureList = {};
 var chart;
 var selectedCells;
 var chartSeries;
 function dataBound() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 if (onInit) {
 for (var i = 0; i <
pivotObj.dataSourceSettings.values.length; i++) {
 var value = pivotObj.dataSourceSettings.values[i];
 measureList[value.name] = value.caption || value.name;
 }
 pivotObj.grid.selectionModule.selectCellsByRange({
cellIndex: 1, rowIndex: 1 }, { cellIndex: 3, rowIndex: 3 });
 }
 }
 function cellSelected(args) {
 selectedCells = args.selectedCellsInfo;
 if (selectedCells && selectedCells.length > 0) {
 chartSeries = frameChartSeries();
 chartUpdate();
 }
 }

```

```

function frameChartSeries() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 var columnGroupObject = {};
 for (var b = 0; b < selectedCells.length; b++) {
 var cell = selectedCells[b];
 if (cell.measure !== '') {
 var columnSeries =
(pivotObj.dataSourceSettings.values.length > 1 && measureList[cell.measure])
?
 (cell.columnHeaders.toString() + ' ~ ' +
measureList[cell.measure]) : cell.columnHeaders.toString();
 if (columnGroupObject[columnSeries]) {
 columnGroupObject[columnSeries].push({ x:
cell.rowHeaders == '' ? 'Grand Total' : cell.rowHeaders.toString(), y:
Number(cell.value) });
 }
 else {
 columnGroupObject[columnSeries] = [{ x:
cell.rowHeaders == '' ? 'Grand Total' : cell.rowHeaders.toString(), y:
Number(cell.value) }];
 }
 }
 }
 var columnKeys = Object.keys(columnGroupObject);
 var chartSeries = [];
 for (var c = 0; c < columnKeys.length; c++) {
 var key = columnKeys[c];
 chartSeries.push({
 dataSource: columnGroupObject[key],
 xName: 'x',
 yName: 'y',
 type: 'Column',
 name: key
 });
 }
 return chartSeries;
}

function chartUpdate() {
 var chart = document.getElementById('Chart').ej2_instances[0];
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 if (onInit) {
 onInit = false;
 chart = new ej.charts.Chart({
 title: 'Sales Analysis',
 legendSettings: {
 visible: true
 },
 tooltip: {
 enable: true
 },
 primaryYAxis: {
 title: pivotObj.dataSourceSettings.values.map(function
(args) { return args.caption || args.name; }).join(' ~ '),
 },
 primaryXAxis: {

```

```

 valueType: 'Category',
 title: pivotObj.dataSourceSettings.rows.map(function
(args) { return args.caption || args.name; }).join(' ~ '),
 labelIntersectAction: 'Rotate45'
 },
 series: chartSeries,
 }, '#Chart');
}
else {
 chart.series = chartSeries;
 chart.primaryXAxis.title =
pivotObj.dataSourceSettings.rows.map(function (args) { return args.caption
|| args.name; }).join(' ~ ');
 chart.primaryYAxis.title =
pivotObj.dataSourceSettings.values.map(function (args) { return args.caption
|| args.name; }).join(' ~ ');
 chart.refresh();
}
}
function onLoad() {
 if (onInit) {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.gridSettings.selectionSettings= { mode: 'Cell',
type: 'Multiple', cellSelectionMode: 'Box' };
 }
}
</script>

```

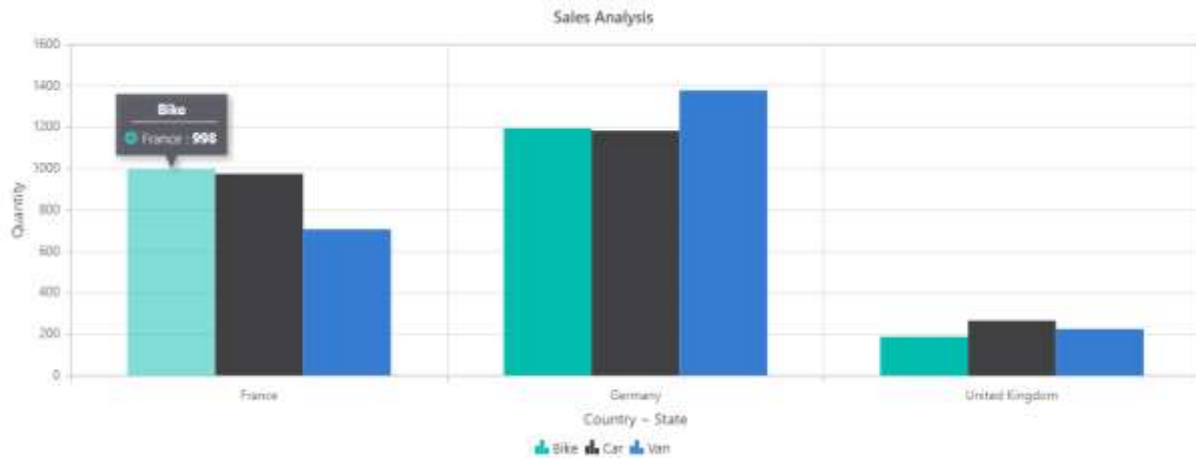
### POPUP.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                  | ► Bike | ► Car | ► Van | Grand Total |
|------------------|--------|-------|-------|-------------|
| ► Canada         | 1070   | 1188  | 923   | 3163        |
| ► France         | 998    | 975   | 707   | 2680        |
| ► Germany        | 1195   | 1183  | 1378  | 3756        |
| ► United Kingdom | 188    | 256   | 226   | 680         |
| ► United States  | 1753   | 1570  | 1547  | 4970        |
| Grand Total      | 5204   | 5162  | 4883  | 15249       |



### Drill-through grid's cell edit type

Using the [DrillThrough](#) event in the pivot table, you can define the edit type of a particular column in the grid present inside the drill-through dialog. To do so, check the column name in the [DrillThrough](#) event and then specify the edit type of that column using the [GridColumn.EditType](#) event argument.

**Note:** The [GridColumn.EditType](#) property must be set based on the column's data type. For example, the string data type will not be applicable for the numeric text box edit type.

- `NumericTextBox` control for integer, double, and decimal data types.
- `TextBox` control for string data type.
- `DropDownList` control to show all unique values related to that field.
- `CheckBox` control for boolean data type.
- `DatePicker` control for date data type.
- `DateTimePicker` control for date time data type.

In the below example, the data type of the `Country` column is set to `DropDownList`.

### CSHTML

```
@Html.EJS().PivotView("PivotGrid").Width("100%").Height("300").DataSource(dataSource =>
dataSource.Data((IEnumerable<object>) ViewBag.Data).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
```

```

 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Production Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).DrillThrough("drillThrough").EditSettings(new
PivotViewCellEditSettings{AllowAdding=true;AllowDeleting=true;AllowEditing=true;AllowCommandColumns=true}).Render()
<script>
function drillThrough(args) {
 for (var i = 0; i < args.gridColumns.length; i++) {
 if (args.gridColumns[i].field === 'Country') {
 args.gridColumns[i].editType = 'dropdownedit';
 //args.gridColumns[i].editType = 'numericedit';
 //args.gridColumns[i].editType = 'textedit';
 //args.gridColumns[i].editType = 'booleanedit';
 //args.gridColumns[i].editType = 'datepickeredit';
 //args.gridColumns[i].editType = 'datetimepickeredit';
 }
 }
}
</script>

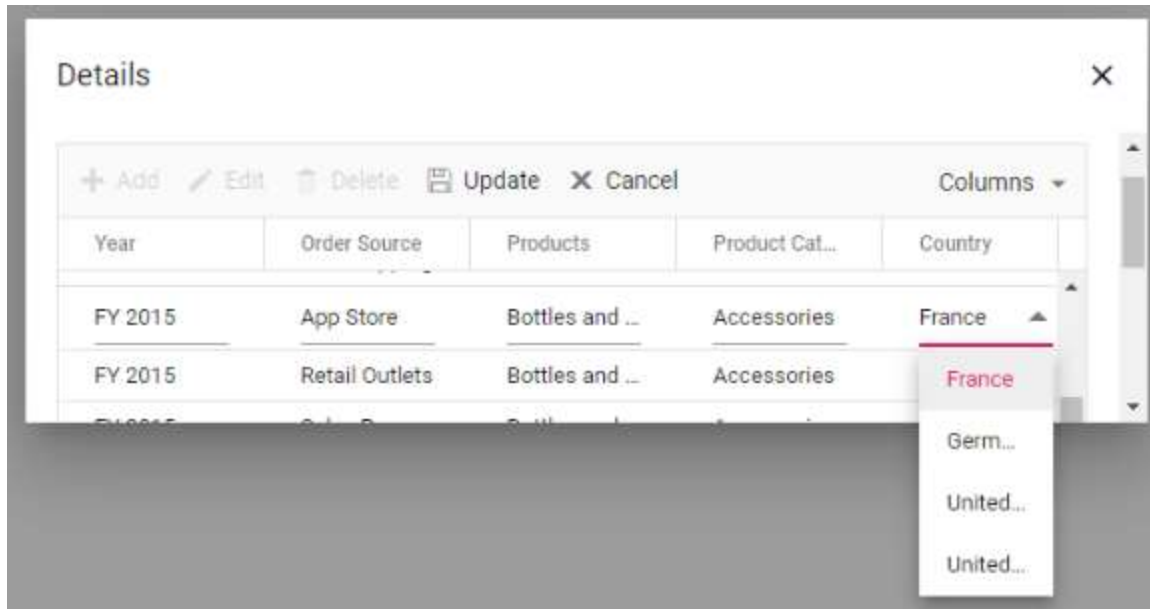
```

### EDITTYPE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```



### Show field list when pivot table is empty

When there are no fields in a pivot table's row, column, value, and filter axes, a field list can still be displayed. To do so, use the [DataBound](#) event and call the `onShowFieldList` method as shown below.

### CSHTML

```
@Html.EJS().PivotView("PivotGrid").Width("100%).Height("300").DataSource(dataSource =>
dataSource.Data((IEnumerable<object>) ViewBag.Data).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatsettings =>
{
formatsettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
}).Rows(rows =>
{
rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
columns.Name("Year").Caption("Production Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).DrillThrough("drillThrough").EditSettings(new
PivotViewCellEditSettings{AllowAdding=true;AllowDeleting=true;AllowEditing=true;AllowCommandColumns=true}).Render()
<script>
function drillThrough(args) {
for (var i = 0; i < args.gridColumns.length; i++) {
if (args.gridColumns[i].field === 'Country') {
args.gridColumns[i].editType = 'dropdownedit';
//args.gridColumns[i].editType = 'numericedit';
```

```

 //args.gridColumns[i].editType = 'textedit';
 //args.gridColumns[i].editType = 'booleanedit';
 //args.gridColumns[i].editType = 'datepickeredit';
 //args.gridColumns[i].editType = 'datetimepickeredit';
 }
}
}
</script>

```

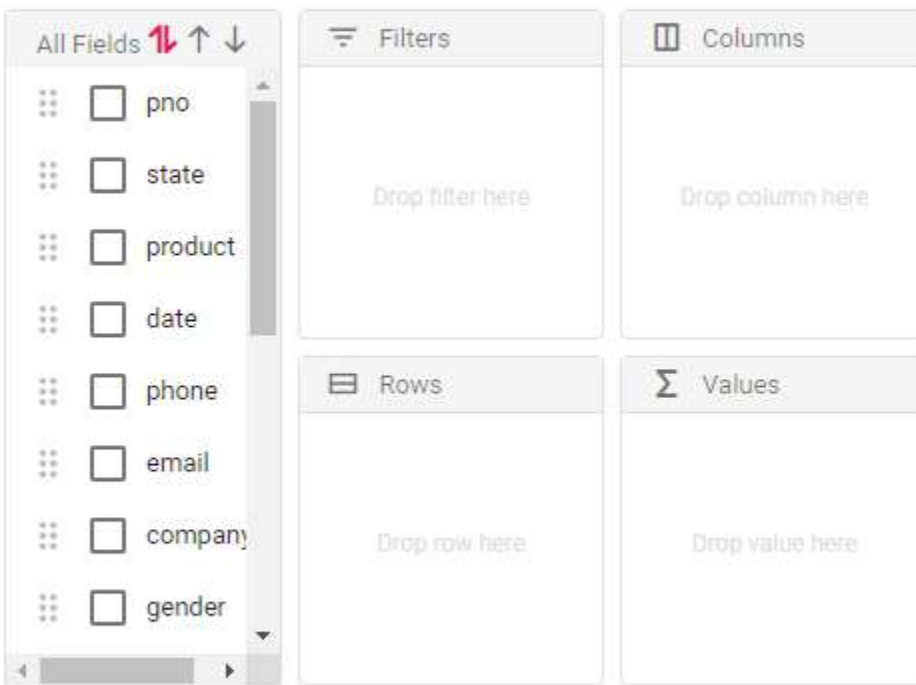
### EDITTYPE.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

### Field List



CLOSE

Apply custom style to pivot cells in ASP.NET MVC Pivot Table Component

The [QueryCellInfo](#) event in [PivotViewGridSettings](#) can be used to apply custom style to row and value cells, and the [HeaderCellInfo](#) event in [PivotViewGridSettings](#) can be used to apply custom styles to column cells.

In the following example, a custom style has been applied to the column header **“Sold Amount”** under **“FY 2016”** via the [HeaderCellInfo](#) event and to the row header **“Germany”** and its aggregated value via the [QueryCellInfo](#) event by adding the **“e-custom”** class to the cell element.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(gridSettings =>
gridSettings.QueryCellInfo("querycell").HeaderCellInfo("headerCellInfo")).Render()
<style>
 .e-custom {
 font-family: 'Courier New', Courier, monospace;
 font-size: 12px !important;
 background-color: pink !important;
 }
</style>
<script>
 function queryCell(args) {
 var colIndex = Number(args.cell.getAttribute('data-colindex'));
 var cells = args.data[colIndex] ? args.data[colIndex] : {};
 // Here by using 'actualText' option, a custom class can be added
 to the specific row header and its value to apply custom style.
 if (cells.actualText === 'Germany') {
 args.cell.classList.add('e-custom');
 } else if (cells.actualText === 'Amount' &&
 cells.columnHeaders === 'FY 2016' && cells.rowHeaders ===
'Germany') {
 args.cell.classList.add('e-custom');
 }
 }
 function headerCellInfo(args) {
 var customAttributes = args.cell.column.customAttributes;
 // Here custom class can be added to the specific column header
 by using unique level name, to apply custom style.
 if (args.node.classList.contains('e-columnsheader') &&
 customAttributes &&
 customAttributes.cell.valueSort.levelName === 'FY 2016.Sold
Amount') {
```



```

 args.node.classList.add('e-custom');
 }
}
</script>

```

### CUSTOMSTYLES.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |             | Grand Total |
|---------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|-------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  |
| France        | 450        | \$714,955   | 526        | \$1,542,104 | 592        | \$2,903,308 | 16         | \$27,264    | 1584        |
| Germany       | 440        | \$563,515   | 496        | \$1,772,104 | 372        | \$1,034,808 | 96         | \$77,264    | 1404        |
| United States | 546        | \$754,515   | 636        | \$2,263,104 | 632        | \$3,041,448 | 76         | \$97,264    | 1890        |
| Grand Total   | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1596       | \$7,579,564 | 188        | \$201,792   | 4878        |

**Note:** The **dot(.)** character in **FY 2016.Sold Amount** is used by default to identify the header levels in the pivot table's row and column. It can be changed by setting the [HeaderDelimiter](#) in the [PivotViewValueSortSettings](#) property to any other delimiter instead of the default separator.

Show tooltip for row and column headers in ASP.NET MVC Pivot Table Component

You can create and display the tooltip for each row and column header(s) in the pivot table by using an external tooltip component via the [DataBound](#) event.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).DataBound("dataBound").Render()
<script>
 function dataBound() {
 var pivotObj =
 document.getElementById('PivotView').ej2_instances[0];

```

```

 var headerTooltip;
 if (!headerTooltip) {
 headerTooltip = new ej.popups.Tooltip({
 target: 'td.e-rowsheader,th.e-columnsheader', beforeRender:
beforeRender
 });
 headerTooltip.appendTo(pivotObj.element);
 }
 }
 function beforeRender(args) {
 var pivotObj =
document.getElementById('PivotView').ej2_instances[0];
 if (args.target.parentElement.querySelector('.e-rowsheader')) {
 // Here you can set custom content for row header(s) tooltip
from its cell information.
 var index = Number(args.target.getAttributeNode('index').value);
 var colIndex = Number(args.target.getAttributeNode('data-
colindex').value);
 var cell = pivotObj.engineModule.pivotValues[index][colIndex];
 var valueText = cell.valueSort ? cell.valueSort : '';
 if (cell.formattedText !== 'Grand Total') {
 this.content =
 '<div>' +
 'FieldName: ' +
 valueText.axis +
 '</br>' +
 'Text: ' +
 cell.formattedText +
 '</div>';
 } else {
 this.content =
 '<div>' +
 'FieldName: ' +
 valueText.uniqueName +
 '</br>' +
 'Text: ' +
 cell.formattedText +
 '</div>';
 }
 } else {
 // Here you can set custom content for column header(s) tooltip
from its cell information.
 if (args.target.querySelector('.e-cellvalue')) {
 this.content = args.target.querySelector('.e-
cellvalue').innerText;
 } else if (args.target.querySelector('.e-headertext')) {
 this.content = args.target.querySelector('.e-
headertext').innerText;
 } else if (args.target.querySelector('.e-stackedheadercelldiv'))
{
 this.content = args.target.querySelector('.e-
stackedheadercelldiv').innerText;
 } else {
 this.content = '';
 }
 }
 }
}

```

```
</script>
```

### CUSTOMTOOLTIP.CS

```
public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}
```

|                      | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |             | Grand Total |              |
|----------------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|-------------|--------------|
|                      | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  | Sold Amount  |
| France               | 450        | \$714,955   | 526        | \$1,542,104 | 592        | \$2,903,306 | 16         | \$27,264    | 1584        | \$5,187,631  |
| FetchHeader: Country | 440        | \$563,515   | 496        | \$1,772,104 | 372        | \$1,634,908 | 96         | \$77,264    | 1404        | \$4,047,691  |
| Text: France         | 546        | \$754,515   | 636        | \$2,263,104 | 632        | \$3,041,448 | 78         | \$97,204    | 1890        | \$6,156,331  |
| Grand Total          | 1436       | \$2,032,985 | 1658       | \$5,577,312 | 1596       | \$7,579,564 | 188        | \$201,792   | 4878        | \$15,391,653 |

### Hide specific columns in ASP.NET MVC Pivot Table Component

By using the [ColumnRender](#) event in the [PivotViewGridSettings](#), you can hide specific column(s) in the pivot table. In the example below, the “Units Sold” column under “FY 2016” is hidden by setting its **visible** property to **false** via the [ColumnRender](#) event.

**Note:** The **dot(.)** character in **FY 2016.Units Sold** is used by default to identify the header levels in the pivot table's row and column. It can be changed by setting the [HeaderDelimiter](#) in the [PivotViewValueSortSettings](#) property to any other delimiter instead of the default separator.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height("300").Width(650).DataSourceSettings(
 dataSourceSettings =>
 {
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false);
 dataSourceSettings.FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {
 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 }).GridSettings(gridSettings =>
 {
 gridSettings.ColumnRender("columnRender").Render()
 })
)
<script>
 function columnRender(args) {
 for (var i = 1; i < args.columns.length; i++) {
```

```

 if (args.stackedColumns[i].customAttributes &&
args.stackedColumns[i].customAttributes.cell.valueSort.levelName === 'FY
2016.Units Sold') {
 args.stackedColumns[i].visible = false;
 }
 }
}
</script>

```

### HIDESPECIFICCOLUMN.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016     | FY 2017    | FY 2018     |            | Grand Total |              |
|---------------|------------|-------------|-------------|------------|-------------|------------|-------------|--------------|
|               | Units Sold | Sold Amount | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Sold Amount  |
| France        | 450        | \$714,955   | \$1,542,104 | 392        | \$2,903,308 | 16         | \$27,264    | \$5,187,631  |
| Germany       | 440        | \$563,515   | \$1,772,104 | 372        | \$1,634,000 | 96         | \$77,264    | \$4,047,691  |
| United States | 546        | \$754,515   | \$2,261,104 | 632        | \$3,041,448 | 76         | \$97,264    | \$6,156,331  |
| Grand Total   | 1436       | \$2,032,985 | \$5,577,312 | 1396       | \$7,578,564 | 188        | \$201,792   | \$15,391,653 |

Export table and chart into the same document using toolbar

Even if the [PivotViewDisplayOption.View](#) property is set to **Both** in the pivot table, you can only export either the table or the chart to the PDF document based on the current value set in the [PivotViewDisplayOption.Primary](#) property. But, to export both the table and the chart to the same PDF document, use the `pdfExport` method during the [ActionBegin](#) event invoke.

In the following example, the built-in export action can be restricted by setting the `args.cancel` option to **true** in the [ActionBegin](#) event, and both the table and the chart can be exported by calling the `pdfExport` method and setting the `exportBothTableAndChart` argument to **true**.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Height("300").ShowToolbar(true).ShowField
List(true).AllowPdfExport(true).DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).Expan
dAll(false)
.FormatSettings(formatsettings =>
{
 formatsettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{

```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).GridSettings(new Syncfusion.EJ2.PivotView.PivotViewGridSettings {
 ColumnWidth = 140 }).DisplayOption(new PivotViewDisplayOption { View =
 Syncfusion.EJ2.PivotView.View.Both }).Toolbar(new List<string>
 () { "Grid", "Chart", "Export", "FieldList"
 }).ActionBegin("actionBegin").Render()
<script>
 function actionBegin(args) {
 var pivotTableObj =
 document.getElementById('pivotview').ej2_instances[0];
 if (args.actionName == 'PDF export') {
 args.cancel = true;
 pivotTableObj.pdfExport({}, false, null, false, true);
 }
 }
</script>

```

### EXPORTTABLEANDCHART.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

<!-- markdownlint-disable MD009 -->

#### Add custom aggregation type to the menu in ASP.NET MVC Pivot Table Component

By using the [DataBound](#) event, you can add your own custom aggregate type(s) to the pivot table's aggregate menu.

In the following example, we have added the aggregation types **CustomAggregateType 1** and **CustomAggregateType 2** to the aggregate menu. The calculation for those aggregated types can be done using the [AggregateCellInfo](#) event.

### CSHTML

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Height("300").DataSourceSettings(dataSourceSettings =>
 dataSourceSettings.DataSource((IEnumerable<object>)ViewBag.DataSource).ExpandAll(false)
 .FormatSettings(formatSettings =>
 {
 formatSettings.Name("Amount").Format("C0").Add();
 }).Rows(rows =>
 {
 rows.Name("Country").Add(); rows.Name("Products").Add();
 }).Columns(columns =>
 {
 columns.Name("Year").Caption("Year").Add();
 columns.Name("Quarter").Add();
 }).Values(values =>
 {

```

```

 values.Name("Sold").Caption("Units Sold").Add();
 values.Name("Amount").Caption("Sold Amount").Add();
 })).GridSettings(new PivotViewGridSettings { ColumnWidth = 140
 }).AggregateCellInfo("aggregateCell").DataBound("dataBound").Render()
<script>
 var SummaryType = [
 'Sum',
 'Count',
 'DistinctCount',
 'Avg',
 'CustomAggregateType1',
 'CustomAggregateType2'
];
 var L10n = ej.base.L10n;
 L10n.load({
 'en-US': {
 pivotview: {
 CustomAggregateType1: 'Custom Aggregate Type 1',
 CustomAggregateType2: 'Custom Aggregate Type 2',
 },
 pivotfieldlist: {
 CustomAggregateType1: 'Custom Aggregate Type 1',
 CustomAggregateType2: 'Custom Aggregate Type 2',
 }
 }
 });
 function dataBound() {
 var pivotObj =
 document.getElementById('pivotview').ej2_instances[0];
 pivotObj.getAllSummaryType = function () {
 return SummaryType;
 };
 pivotObj.pivotFieldListModule.aggregateTypes = SummaryType;
 pivotObj.pivotFieldListModule.getAllSummaryType = function () {
 return SummaryType;
 };
 }
 function aggregateCell(args) {
 if (args.aggregateType === 'CustomAggregateType1') {
 args.value = args.value * 100;
 }
 if (args.aggregateType === 'CustomAggregateType2') {
 args.value = args.value / 100;
 }
 }
}
</script>

```

### CUSTOMAGGREGATION.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                 | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |             | Grand Total |             |
|-----------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|-------------|-------------|
|                 | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  | Sold Amount |
| > France        | 45000      | \$7,150     | 52600      | \$15,421    | 59200      | \$29,033    | 1600       | \$273       | 158400      | \$51,876    |
| > Germany       | 44000      | \$5,635     | 49600      | \$17,721    | 37200      | \$16,348    | 9600       | \$773       | 140400      | \$40,477    |
| > United States | 54600      | \$7,545     | 63600      | \$22,631    | 63200      | \$30,414    | 7600       | \$973       | 189000      | \$61,563    |
| Grand Total     | 143600     | \$20,330    | 165800     | \$55,773    | 159600     | \$75,796    | 18800      | \$2,018     | 487800      | \$153,917   |

<!-- markdownlint-disable MD009 -->

### Convert complex JSON to flat JSON and assign it to the pivot table

By default, flat JSON can only bind to the pivot table. However, you can connect complex JSON to the pivot table by converting it to flat JSON via code-behind and binding it to the pivot table using the [DataSource](#) property in the [Load](#) event.

In the following example, the **complexToFlatJson()** method is used to convert complex JSON to flat JSON and bind it to the pivot table using the [DataSource](#) property, then modifying the field names in the [Rows](#) and [Columns](#) based on the converted flat JSON under [PivotViewDataSourceSettings](#) in the [Load](#) event.

### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("PivotView").Height(300).DataSourceSettings(dataSource =>
 dataSource.DataSource(data()).ExpandAll(false)
 .ValueSortSettings(new PivotViewValueSortSettings {
 HeaderDelimiter = "-",
 }).Rows(rows =>
 {
 rows.Name("ShipDetails").Add();
 }).Columns(columns =>
 {
 columns.Name("OrderDetails").Add();
 }).Values(values =>
 {
 values.Name("Freight").Caption("Units Sold").Add();
 })).Load("load").Render()
<script>
 function load(args) {
 dataSource =
 JSON.parse(JSON.stringify(args.dataSourceSettings.dataSource));
 args.dataSourceSettings.dataSource = complexToFlatJson(dataSource);
 var rows = [];
 for (var i = 0; i < args.dataSourceSettings.rows.length; i++) {
 if (args.dataSourceSettings.rows[i].name in parentProp) {
 rows =
 rows.concat(parentProp[args.dataSourceSettings.rows[i].name]);
 }
 else {
 rows.push(args.dataSourceSettings.rows[i]);
 }
 }
 args.dataSourceSettings.rows = rows;
 var columns = [];
 for (var i = 0; i < args.dataSourceSettings.columns.length; i++) {
 if (args.dataSourceSettings.columns[i].name in parentProp) {
 columns =
 columns.concat(parentProp[args.dataSourceSettings.columns[i].name]);
 }
 }
 }
}
```

```

 else {
 columns.push(args.dataSourceSettings.columns[i]);
 }
 }
 args.dataSourceSettings.columns = columns;
}
function complexToFlatJson(data) {
 var flatArray = [];
 var flatObject = {};
 for (var index = 0; index < data.length; index++) {
 for (var prop in data[index]) {
 var value = data[index][prop];
 if (Array.isArray(value)) {
 for (var i = 0; i < value.length; i++) {
 var childProp = [];
 for (var inProp in value[i]) {
 flatObject[inProp] = value[i][inProp];
 var object = {
 name: inProp,
 };
 childProp.push(object);
 }
 parentProp[prop] = childProp;
 }
 }
 else {
 flatObject[prop] = value;
 }
 }
 flatArray.push(flatObject);
 flatObject = {};
 }
 return flatArray;
}
var data = function () {
 return [
 {
 CustomerID: 'VINET',
 Freight: 32.38,
 OrderDetails: [
 {
 OrderID: 10248,
 OrderDate: '1996-07-04T10:10:00.000Z',
 }
],
 ShipDetails: [
 {
 ShipName: 'Vins et alcools Chevalier',
 ShipAddress: '59 rue de l'Abbaye',
 ShipCity: 'Reims',
 ShipRegion: null,
 ShipCountry: 'France',
 ShippedDate: '1996-07-16T12:20:00.000Z',
 }
]
 },
 {

```



```

CustomerID: 'GALED',
Freight: 10.14,
OrderDetails: [
 {
 OrderID: 10366,
 OrderDate: '1996-11-28T00:00:00.000Z',
 }
],
ShipDetails: [
 {
 ShippedDate: '1996-12-30T00:00:00.000Z',
 ShipName: 'Galería del gastronómo',
 ShipAddress: 'Rambla de Cataluña, 23',
 ShipCity: 'Barcelona',
 ShipRegion: null,
 ShipCountry: 'Spain',
 }
]
},
{
 CustomerID: 'VAFFE',
 Freight: 13.55,
 OrderDetails: [
 {
 OrderID: 10367,
 OrderDate: '1996-12-02T00:00:00.000Z',
 }
],
 ShipDetails: [
 {
 ShippedDate: '1996-12-30T00:00:00.000Z',
 ShipName: 'Vaffeljernet',
 ShipAddress: 'Smagsloget 45',
 ShipCity: 'Århus',
 ShipRegion: null,
 ShipCountry: 'Denmark',
 }
]
},
{
 CustomerID: 'ERNSH',
 Freight: 101.95,
 OrderDetails: [
 {
 OrderID: 10368,
 OrderDate: '1996-11-29T00:00:00.000Z',
 }
],
 ShipDetails: [
 {
 ShippedDate: '1996-12-30T00:00:00.000Z',
 ShipName: 'Ernst Handel',
 ShipAddress: 'Kirchgasse 6',
 ShipCity: 'Graz',
 ShipRegion: null,
 ShipCountry: 'Austria',
 }
]
}

```

```

],
 {
 CustomerID: 'SPLIR',
 Freight: 195.68,
 OrderDetails: [
 {
 OrderID: 10369,
 OrderDate: '1996-11-28T00:00:00.000Z',
 }
],
 ShipDetails: [
 {
 ShippedDate: '1996-12-30T00:00:00.000Z',
 ShipName: 'Split Rail Beer & Ale',
 ShipAddress: 'P.O. Box 555',
 ShipCity: 'Lander',
 ShipRegion: 'WY',
 ShipCountry: 'USA',
 }
],
 }
];
};
</script>

```

### COMPLEXTOFLATJSON.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|                          | 10248                    | 10366                    | 10367                    | 10368                    |
|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
|                          | 1996-07-04T10:10:00.000Z | 1996-11-28T00:00:00.000Z | 1996-12-02T00:00:00.000Z | 1996-11-29T00:00:00.000Z |
| 1996-07-16T12:20:00.000Z | 32.38                    |                          |                          |                          |
| Vins et alcools Cheva... | 32.38                    |                          |                          |                          |
| 59 rue de l'Abbaye       | 32.38                    |                          |                          |                          |
| Reims                    | 32.38                    |                          |                          |                          |
| null                     | 32.38                    |                          |                          |                          |
| France                   | 32.38                    |                          |                          |                          |
| 1996-12-30T00:00:00.000Z |                          | 10.14                    | 13.55                    |                          |
| Ernst Handel             |                          |                          |                          |                          |
| Kirchgasse 6             |                          |                          |                          |                          |
| Graz                     |                          |                          |                          |                          |
| null                     |                          |                          |                          |                          |
| Austria                  |                          |                          |                          |                          |
| Galeria del gastronóm... |                          | 10.14                    |                          |                          |
| Rambla de Catalu...      |                          | 10.14                    |                          |                          |

<!-- markdownlint-disable MD009 -->

### Load desired report from the report list as default in ASP.NET MVC Pivot Table Component

By default, the pivot table is displayed with the report bound at the code-behind. To load a desired report from the previously saved report collection during initial rendering, set the desired report name in the [DataBound](#) event, along with the additional report-based customization code shown below.

#### CSHTML

```
@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Width("100%").Height("300").ShowToolBar(true).ShowFieldList(true).AllowExcelExport(true).AllowNumberFormatting(true).AllowConditionalFormatting(true).AllowPdfExport(true).ShowToolBar(true).AllowCalculatedField(true).DataSourceSettings(dataSource =>
dataSource.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false).EnableSorting(true)
.FormatSettings(formatSettings => {
formatSettings.Name("Amount").Format("C0").MaximumSignificantDigits(10).MinimumSignificantDigits(1).UseGrouping(true).Add(); })
.Rows(rows => { rows.Name("Country").Add(); rows.Name("Products").Add(); })
.Columns(columns => { columns.Name("Year").Add();
columns.Name("Order_Source").Caption("Order Source").Add(); })
.Values(values =>
{
values.Name("In_Stock").Caption("In Stock").Add();
values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
}))
.Filters(filters =>
{
filters.Name("Product_Categories").Caption("Product Categories").Add();
}))
.GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).DisplayOption(new PivotViewDisplayOption { View = View.Both
}).ToolBar(new List<string>
() { "New", "Save", "SaveAs", "Rename", "Remove", "Load", "Grid", "Chart",
"Export", "SubTotal", "GrandTotal", "ConditionalFormatting",
"NumberFormatting" "FieldList"
}).SaveReport("saveReport").LoadReport("loadReport").FetchReport("fetchReport").RenameReport("renameReport").RemoveReport("removeReport").NewReport("newReport").DataBound("dataBound").Render()
<script>
var isInitial = true;
function saveReport(args) {
var reports = [];
var isSaved = false;
if (localStorage.pivotviewReports && localStorage.pivotviewReports !== "") {
reports = JSON.parse(localStorage.pivotviewReports);
}
if (args.report && args.reportName && args.reportName !== '') {
reports.map(function (item) {
if (args.reportName === item.reportName) {
item.report = args.report;
isSaved = true;
}
});
if (!isSaved) {
reports.push(args);
}
}
```

```

 localStorage.pivotviewReports = JSON.stringify(reports);
 }
}
function fetchReport(args) {
 var reportCollection = [];
 var reeportList = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 reeportList.push(item.reportName);
 });
 args.reportName = reeportList;
}
function removeReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 for (var i = 0; i < reportCollection.length; i++) {
 if (reportCollection[i].reportName === args.reportName) {
 reportCollection.splice(i, 1);
 }
 }
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function renameReport(args) {
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 item.reportName = args.rename;
 }
 });
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
!= "") {
 localStorage.pivotviewReports =
JSON.stringify(reportCollection);
 }
}
function newReport() {
 var pivotObj =
document.getElementById('pivotview').ej2_instances[0];
 pivotObj.setProperties({
 dataSourceSettings: {
 columns: [],
 rows: [],
 values: [],

```

```

 filters: []
 }
}, false);
}
function dataBound() {
 var pivotObj = document.getElementById('pivotview').ej2_instances[0]
 if (pivotObj && isInitial) {
 isInitial = false;
 pivotObj.toolbarModule.action = 'Load';
 pivotObj.toolbarModule.reportList.value = 'Default report';
 loadReport({ reportName: 'Default report' });
 }
}
function loadReport(args) {
 var pivotObj = document.getElementById('pivotview').ej2_instances[0]
 var reportCollection = [];
 if (localStorage.pivotviewReports && localStorage.pivotviewReports
 !== "") {
 reportCollection = JSON.parse(localStorage.pivotviewReports);
 }
 reportCollection.map(function (item) {
 if (args.reportName === item.reportName) {
 args.report = item.report;
 }
 });
 if (args.report) {
 pivotObj.dataSourceSettings =
 JSON.parse(args.report).dataSourceSettings;
 }
}
</script>

```

### LOADPREDEFINEDREPORT.CS

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|             | Mountain Bikes |              | Road Bikes |             | Grand Total |              |
|-------------|----------------|--------------|------------|-------------|-------------|--------------|
|             | Sold           | Amount       | Sold       | Amount      | Sold        | Amount       |
| > Q1        | 717            | \$1,140,688  | 565        | \$2,206,690 | 1282        | \$3,347,378  |
| > Q2        | 822            | \$2,200,336  | 622        | \$3,368,043 | 1444        | \$5,568,379  |
| > Q3        | 581            | \$1,300,360  | 1200       | \$3,122,744 | 1781        | \$4,423,104  |
| > Q4        | 2536           | \$6,315,224  | 2342       | \$9,076,429 | 4878        | \$15,391,653 |
| Grand Total | 4878           | \$15,391,653 | 2342       | \$9,076,429 | 7220        | \$24,468,082 |

### Display string value to pivot table values

End user can display string value to the pivot table's value cell by using the [AggregateCellInfo](#) event.

In the following example, each cell value of the **Sold** field's actual value has been assigned from its combination data sets obtained from the `args.cellSets` in the [AggregateCellInfo](#) event.

**CSHTML**

```

@using Syncfusion.EJ2.PivotView
@Html.EJS().PivotView("pivotview").Height("300").DataSourceSettings(dataSourceSettings =>
dataSourceSettings.DataSource((IEnumerable<object>) ViewBag.DataSource).ExpandAll(false)
.FormatSettings(formatSettings =>
{
 formatSettings.Name("Amount").Format("C0").Add();
}).Rows(rows =>
{
 rows.Name("Country").Add(); rows.Name("Products").Add();
}).Columns(columns =>
{
 columns.Name("Year").Caption("Year").Add();
columns.Name("Quarter").Add();
}).Values(values =>
{
 values.Name("Sold").Caption("Units Sold").Add();
values.Name("Amount").Caption("Sold Amount").Add();
})).GridSettings(new PivotViewGridSettings { ColumnWidth = 140
}).AggregateCellInfo("aggregateCell").Render()
<script>
 function aggregateCell(args){
 if (args.fieldName === 'Sold') {
 args.value = secondsToHms(args.value);
 }
 }
 function secondsToHms(d) {
 d = Number(d);
 var h = Math.floor(d / 3600);
 var m = Math.floor((d % 3600) / 60);
 var s = Math.floor((d % 3600) % 60);
 return (
 ('0' + h).slice(-2) + ':' + ('0' + m).slice(-2) + ':' + ('0' +
s).slice(-2)
);
 }
</script>

```

**DISPLAYSTRINGVALUE.CS**

```

public ActionResult Index()
{
 var data = GetPivotData();
 ViewBag.DataSource = data;
 return View();
}

```

|               | FY 2015    |             | FY 2016    |             | FY 2017    |             | FY 2018    |             | Grand Total |              |
|---------------|------------|-------------|------------|-------------|------------|-------------|------------|-------------|-------------|--------------|
|               | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold | Sold Amount | Units Sold  | Sold Amount  |
| France        | 00:07:30   | \$714,855   | 00:08:46   | \$1,543,104 | 00:09:52   | \$2,903,308 | 00:00:16   | \$27,264    | 00:26:24    | \$5,187,631  |
| Germany       | 00:07:20   | \$563,515   | 00:08:16   | \$1,772,104 | 00:06:12   | \$1,634,808 | 00:01:36   | \$77,264    | 00:23:24    | \$4,047,691  |
| United States | 00:09:06   | \$754,515   | 00:10:36   | \$2,263,104 | 00:10:32   | \$3,041,440 | 00:01:16   | \$97,264    | 00:31:30    | \$6,156,331  |
| Grand Total   | 00:23:56   | \$2,032,985 | 00:27:38   | \$5,577,312 | 00:26:36   | \$7,579,564 | 00:03:08   | \$201,792   | 01:21:18    | \$15,391,653 |

## Predefined Dialogs

### Getting Started with ASP.NET Core Predefined Dialogs

This section briefly explains about how to include ASP.NET MVC Predefined Dialogs in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

## Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

## Open ASP.NET MVC Predefined Dialogs

Now, add the Syncfusion ASP.NET MVC Predefined Dialogs in `~/Views/Home/Index.cshtml` page.

Once you completed the setup, you can open predefined dialogs from any where in application using `Alert`, `Confirm` or `Prompt` methods in `DialogUtility`.

*Show alert dialog*

An alert dialog box used to display an errors, warnings, and information alerts that needs user awareness. This can be achieved by using the `DialogUtility.alert` method. The alert dialog is displayed along with the `OK` button. When user clicks on `OK` button, alert dialog will get closed.

In the below code example, alert dialog displayed on button click action.

CSHTML

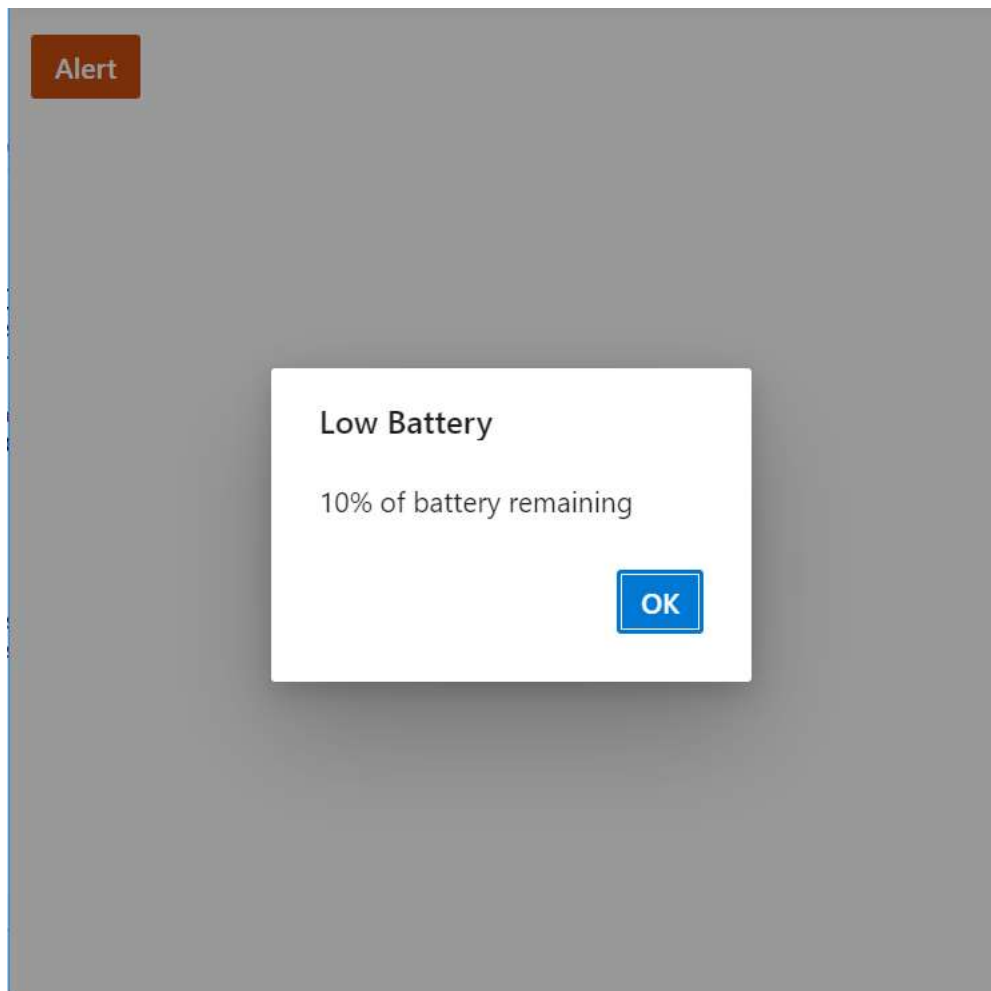
```
<div style="height:400px;">
 <div id="predefinedDialogDefault" class="col-lg-12 control-section">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()

 </div>
</div>
<script>
 var dialogObj;
 document.getElementById('alertBtn').onclick = function () {
 document.getElementById("statusText").style.display = "none";
```



```
dialogObj = ej.popups.DialogUtility.alert({
 title: "Low battery",
 content: "10% of battery remaining",
 okButton: { click: alertBtnClick.bind(this) },
 position: { X: "center", Y: "center" }
});
};
function alertBtnClick() {
 dialogObj.hide();
 document.getElementById("statusText").innerHTML = "The user closed
the Alert dialog.";
 document.getElementById("statusText").style.display = "block";
}
</script>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC predefined dialogs will be rendered in the default web browser.



#### [Show confirm dialog](#)

A confirm dialog box used to displays a specified message along with the **OK** and **Cancel** buttons. This can be achieved by using the `DialogUtility.confirm` method. It is used to get approval from the user,

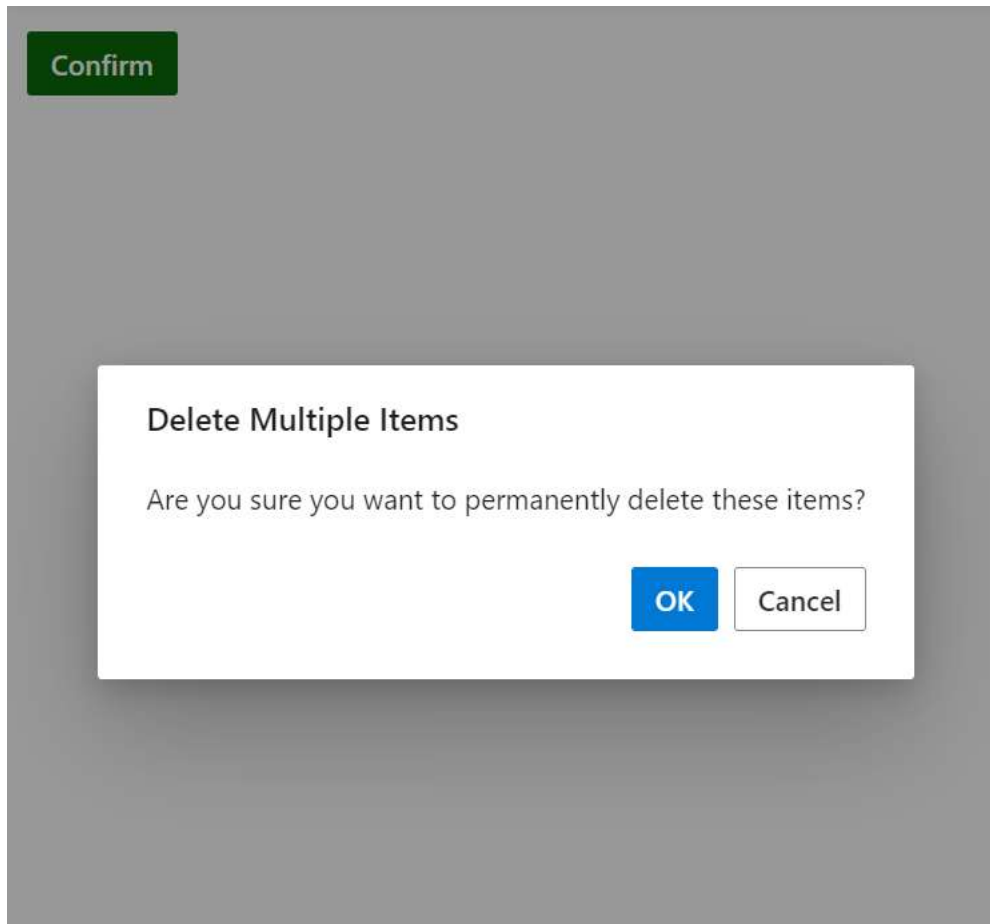
and it appears before any critical action. After get approval from the user the dialog will disappear automatically.

In the below code example, the confirm dialog displayed on **OK** and **Cancel** button click action.

### CSHTML

```
<div style="height:400px;">
 <div id="predefinedDialogDefault" class="col-lg-12 control-section">
 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
success").Render()

 </div>
</div>
<script>
 var dialogObj;
 document.getElementById('confirmBtn').onclick = function () {
 document.getElementById("statusText").style.display = "none";
 dialogObj = ej.popups.DialogUtility.confirm({
 title: "Delete multiple items",
 content: "Are you sure you want to permanently delete these
items?",
 okButton: { click: confirmOkAction.bind(this) },
 cancelButton: { click: confirmCancelAction.bind(this) },
 position: { X: "center", Y: "center" }
 });
 };
 var confirmOkAction = function () {
 dialogObj.hide();
 document.getElementById("statusText").innerHTML = " The user
confirmed the dialog box";
 document.getElementById("statusText").style.display = "block";
 };
 var confirmCancelAction = function () {
 dialogObj.hide();
 document.getElementById("statusText").innerHTML = "The user canceled
the dialog box.";
 document.getElementById("statusText").style.display = "block";
 };
</script>
```



#### Show Prompt dialog

A prompt dialog is used to get the input from the user. When the user clicks the **OK** button the input value from the dialog is returned. If the user clicks the **Cancel** button the null value is returned. After getting the input from the user the dialog will disappear automatically.

In the below code example, the confirm dialog displayed on **OK** and **Cancel** button click action.

#### CSHTML

```
<div style="height:400px;">
 <div id="predefinedDialogDefault" class="col-lg-12 control-section">

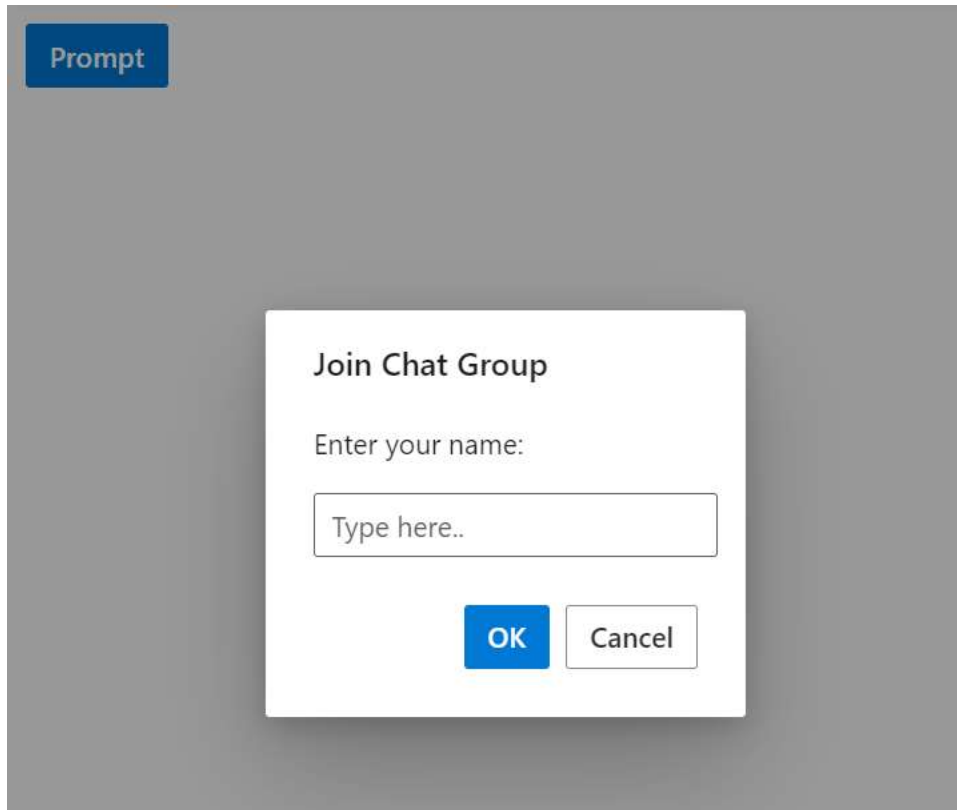
@Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()

 </div>
</div>
<script>
 var dialogObj;
 document.getElementById('promptBtn').onclick = function () {
 document.getElementById("statusText").style.display = "none";
 dialogObj = ej.popups.DialogUtility.confirm({
 title: "Join chat group",
 content: "<p>Enter your name: </p><input id= inputEle type=text
name=Required class=e-input placeholder=Type here.. />",
 okButton: { click: promptOkAction.bind(this) },
 cancelButton: { click: promptCancelAction.bind(this) },
 });
 };
</script>
```

```
 position: { X: "center", Y: "center" }
 });
};
function promptOkAction() {
 var value;
 value = document.getElementById("inputEle").value;
 if (value == "") {
 dialogObj.hide();
 document.getElementById("statusText").innerHTML = "The user's
input is returned as \" \" ";
 document.getElementById("statusText").style.display = "block";
 }
 else {
 dialogObj.hide();
 document.getElementById("statusText").innerHTML = "The user's
input is returned as" + " " + value;
 document.getElementById("statusText").style.display = "block";
 }
}
function promptCancelAction() {
 dialogObj.hide();
 document.getElementById("statusText").innerHTML = "The user canceled
the prompt dialog";
 document.getElementById("statusText").style.display = "block";
}
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}
```



See also

- [Real time example using Dialog](#)
- [Load dialog content using AJAX](#)
- [How to position the dialog on center of the page on scrolling](#)
- [Prevent closing of modal dialog](#)
- [Close dialog while click on outside of dialog](#)
- [How to make a reusable alert and confirm dialog](#)

### Draggable in Predefined Dialogs

The predefined dialogs supports dragging within its target container by grabbing the dialog header, which allows the user to reposition the dialog dynamically by using the `isDraggable` property.

#### Alert

##### ALERT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Low battery",
 content: "10% of battery remaining",
 isDraggable: true
 });
 };
</script>
```

```
};
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Alert()
 {
 return View();
 }
}
```

### Confirm

#### CONFIRM.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
success").Render()
</div>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.confirm({
 title: "Delete multiple items",
 content: "Are you sure you want to permanently delete these
items?",
 isDraggable: true
 });
 };
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}
```

### Prompt

#### PROMPT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()
</div>
<script>
 document.getElementById('promptBtn').onclick = function () {
 ej.popups.DialogUtility.confirm({
 title: "Join chat group",

```

```

 content: "<p>Enter your name: </p> <input id= inputEle type=text
name=Required class=e-input placeholder=Type here.. />",
 isDraggable: true
 });
};
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}

```

### Animation in Predefined Dialogs

The predefined dialogs can be animated during the open and close actions. You can customize the Delay, Duration, and Effect of animation by using the animationSettings property.

In the following sample, the zoom effect is enabled. So, the dialog will open with the zoom in and close with the zoom out effect.

#### Alert

### ALERT.CSHTML

```

<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Low battery",
 content: "10% of battery remaining",
 animationSettings: { effect: 'Zoom', duration: 400 }
 });
 };
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Alert()
 {
 return View();
 }
}

```

#### Confirm

**CONFIRM.CSHTML**

```
<div style="height:400px;">
 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
success").Render()
</div>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Delete multiple items",
 content: "Are you sure you want to permanently delete these
items?",
 animationSettings: { effect: 'Zoom', duration: 400 }
 });
 };
</script>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}
```

**Prompt****PROMPT.CSHTML**

```
<div style="height:400px;">

@Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()
</div>
<script>
 document.getElementById('promptBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Join Wi-Fi network",
 content: "<p>Enter your name: </p><input id=inputEle type=text
name=Required class=e-input placeholder=Type here.. />",
 animationSettings: { effect: 'Zoom', duration: 400 }
 });
 };
</script>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}
```



## Positioning in Predefined Dialogs in Blazor

Customize the dialog position by using the `position` property. The position can be represented with specific `X` and `Y` values.

- The `Position.X` can be configured with a left, center, right or offset value. By default, the value is set as `center`.
- The `Position.Y` can be configured with a top, center, bottom or offset value. By default, the value is set as `center`.

Use the following code snippet for `alert.cshtml`, `confirm.cshtml` and `prompt.cshtml` to customize the position. Here, customized the dialog position as `X= "top"` and `Y= "center"`.

### Alert

#### ALERT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Low battery",
 content: "10% of battery remaining",
 position: { X: "top", Y: "center" }
 });
 };
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Alert()
 {
 return View();
 }
}
```

### Confirm

#### CONFIRM.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
success").Render()
</div>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Delete multiple items",
```

```

 content: "Are you sure you want to permanently delete these
items?",
 position: { X: "top", Y: "center" }
 });
};
</script>

```

**CONTROLLER.CS**

```

public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}

```

**Prompt****PROMPT.CSHTML**

```

<div style="height:400px;">

@Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()
</div>
<script>
 document.getElementById('promptBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Join chat group",
 content: "<p>Enter your name: </p> <input id=inputEle type=text
name=Required class=e-input placeholder=Type here.. />",
 position: { X: "top", Y: "center" }
 });
 };
</script>

```

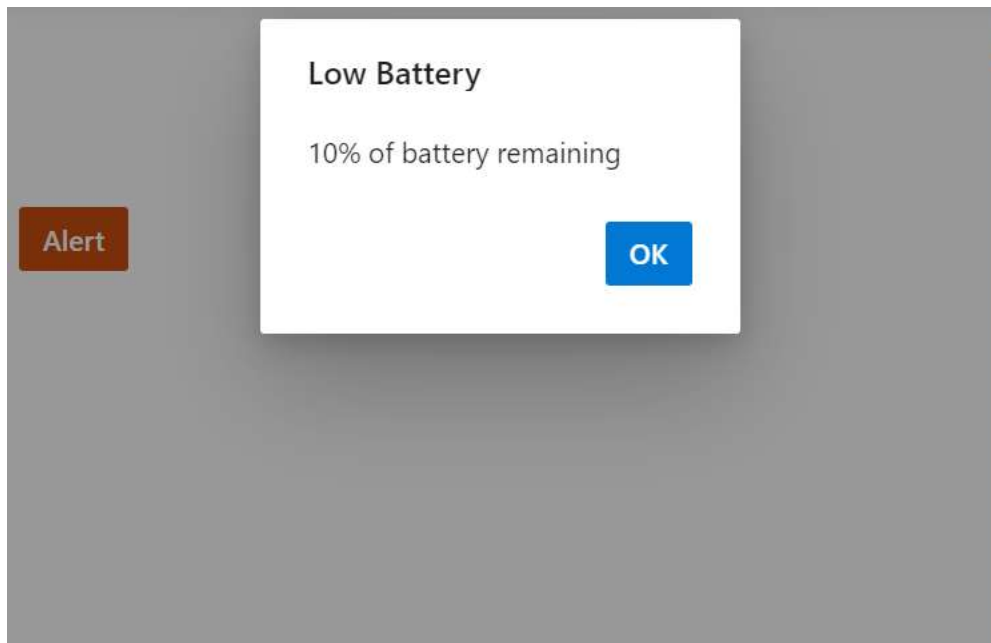
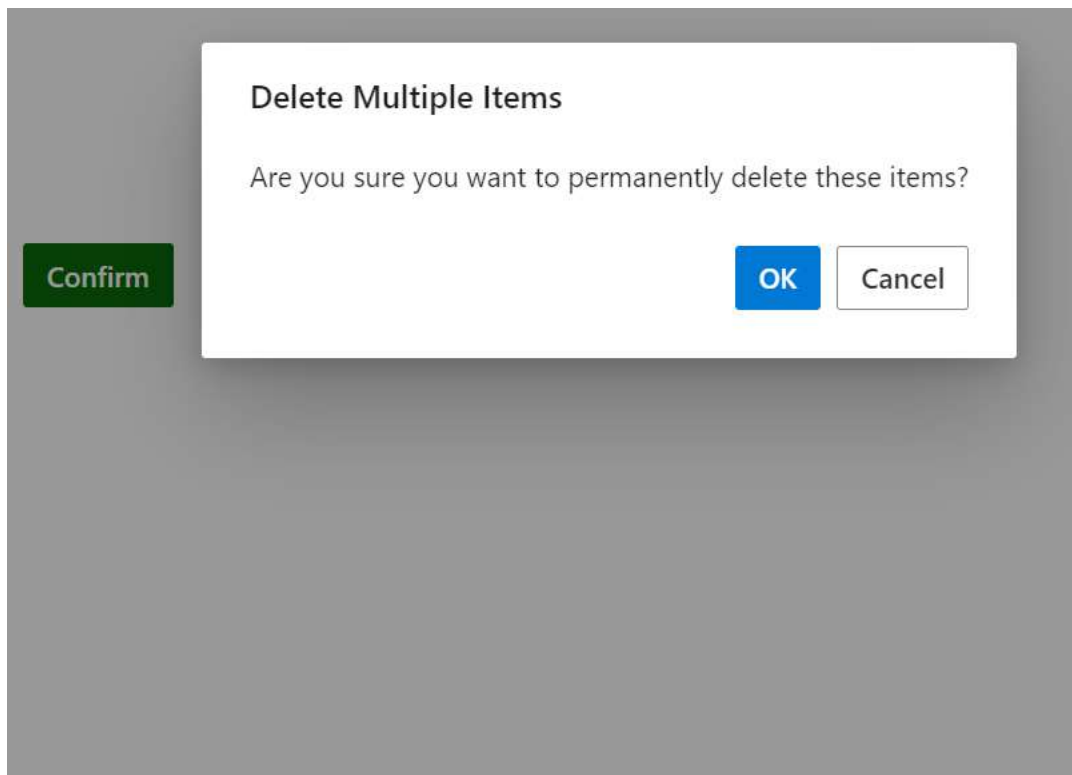
**CONTROLLER.CS**

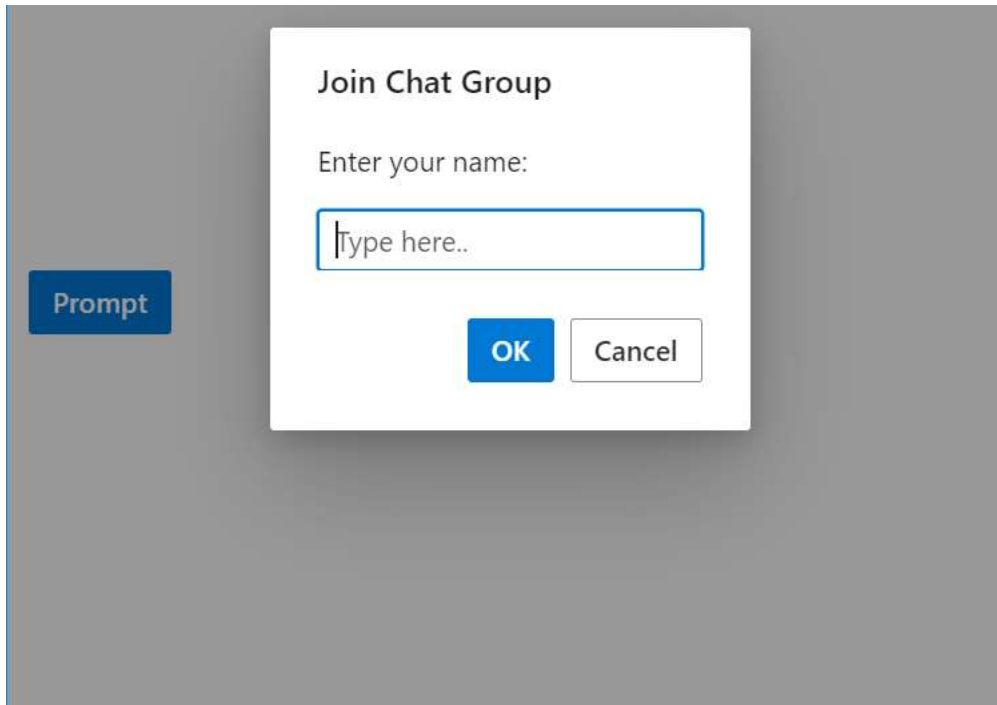
```

public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}

```

**Results from the code snippet****Alert**

**Confirm****Prompt**



### Dimension in Predefined Dialogs

Customize the predefined dialogs dimensions using the `height` and `width` properties.

By default, the predefined dialogs `width` and `height` property value is set as `auto`. Depends on the dialog content the width and height values are automatically adjust. You can specify the dimension values in both pixels and percentage format to change the default dialog width and height values.

Use the following code snippet for **Alert.cshtml**, **Confirm.cshtml** and **Prompt.cshtml** to customize the dialog dimensions.

#### Alert

##### ALERT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Not enough space",
 content: "Delete certain files to free up space to store more
items.",
 width : "300px",
 height : "200px"
 });
 };
</script>
```

##### CONTROLLER.CS

```
public class HomeController : Controller
```

```
{
 public ActionResult Alert()
 {
 return View();
 }
}
```

## Confirm

### CONFIRM.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
success").Render()
</div>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Delete multiple items",
 content: "Are you sure you want to permanently delete these
items?",
 width : "400px",
 height : "200px"
 });
 };
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}
```

## Prompt

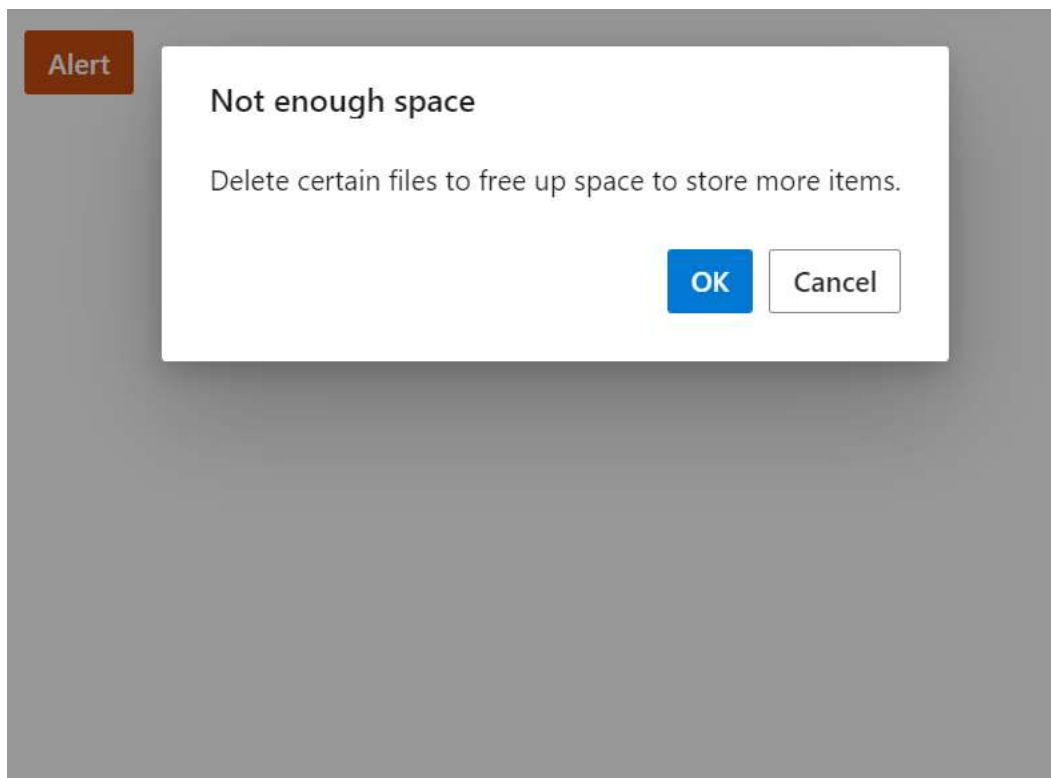
### PROMPT.CSHTML

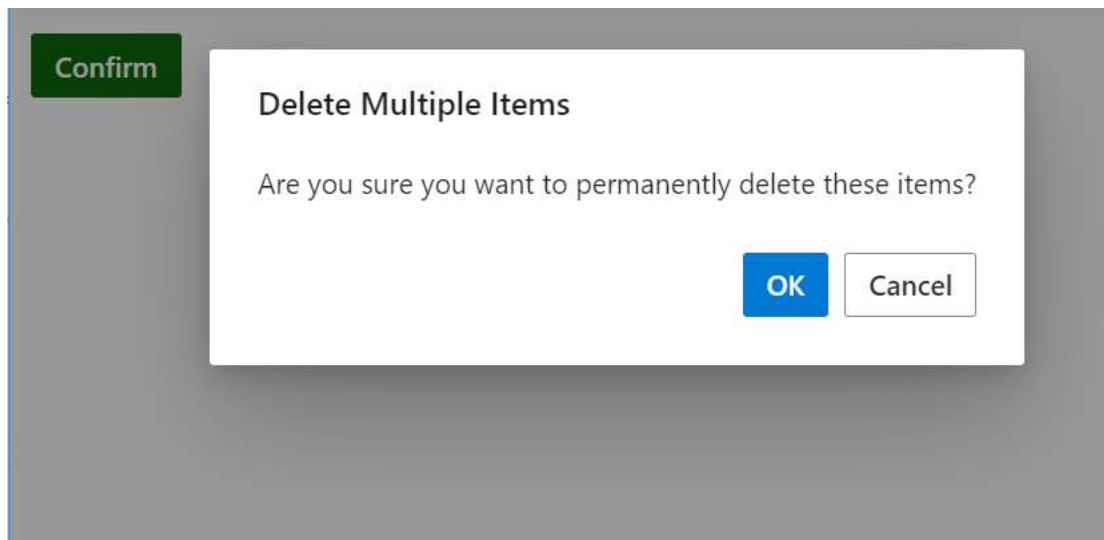
```
<div style="height:400px;">

@Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()
</div>
<script>
 document.getElementById('promptBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Join chat group",
 content: "<p>Enter your name: </p> <input id= inputEle type=text
name=Required class=e-input placeholder=Type here.. />",
 width : "350px",
 height : "210px"
 });
 };
</script>
```

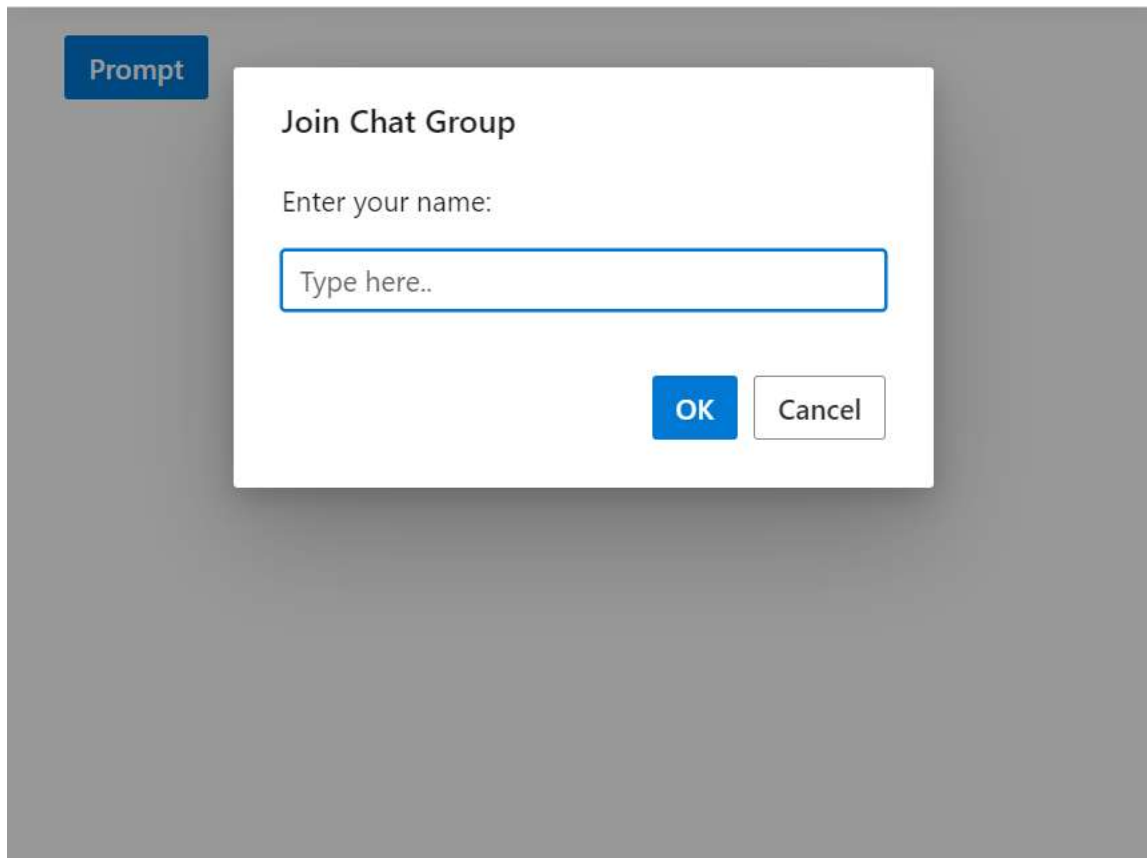
**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}
```

**Results from the code snippet****Alert****Confirm**



### Prompt



### Max-width and max-height

To have a restricted max-width and max-height dialog dimension, you need to specify the max-width, max-height CSS properties for the component's container element by using the `cssClass` property. The max-height value is calculated in source level and set to the dialog. so, need to override the max-height property.

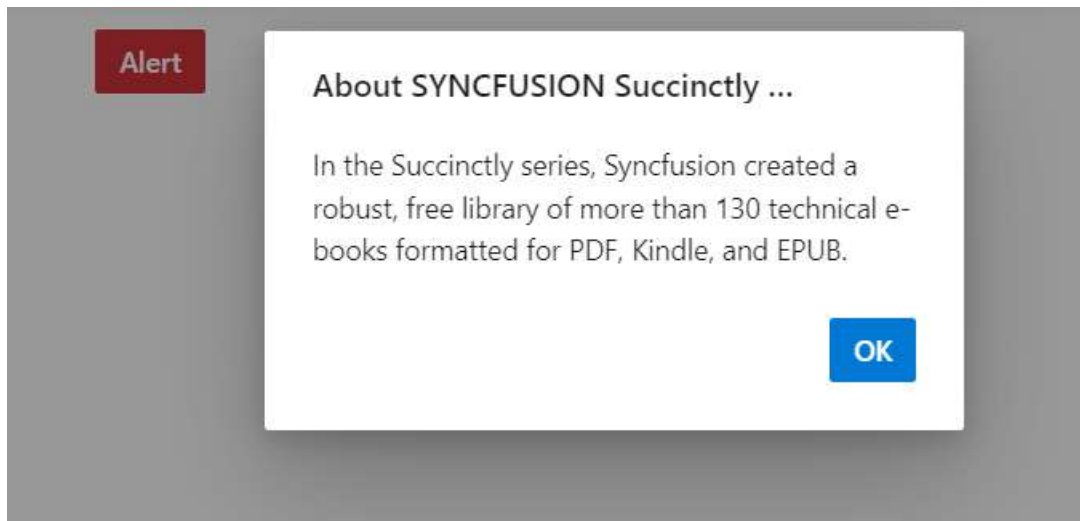
Use the following code to customize the max-width and max-height for alert dialog:

**ALERT.CSHTML**

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<style>
 .e-alert-dialog.customClass {
 max-width: 350px;
 max-height: 250px !important;
 }
</style>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "About syncfusion succinctly series",
 content: "In the succinctly series, syncfusion created a robust,
free library of more than 130 technical e-books formatted for PDF, Kindle,
and EPUB.",
 cssClass: "customClass"
 });
 };
</script>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Alert()
 {
 return View();
 }
}
```

**Min-width and min-height**

To have a restricted min-width and min-height dialog dimension, you need to specify the min-width, min-height CSS properties for the component's container element by using the `cssClass` property.



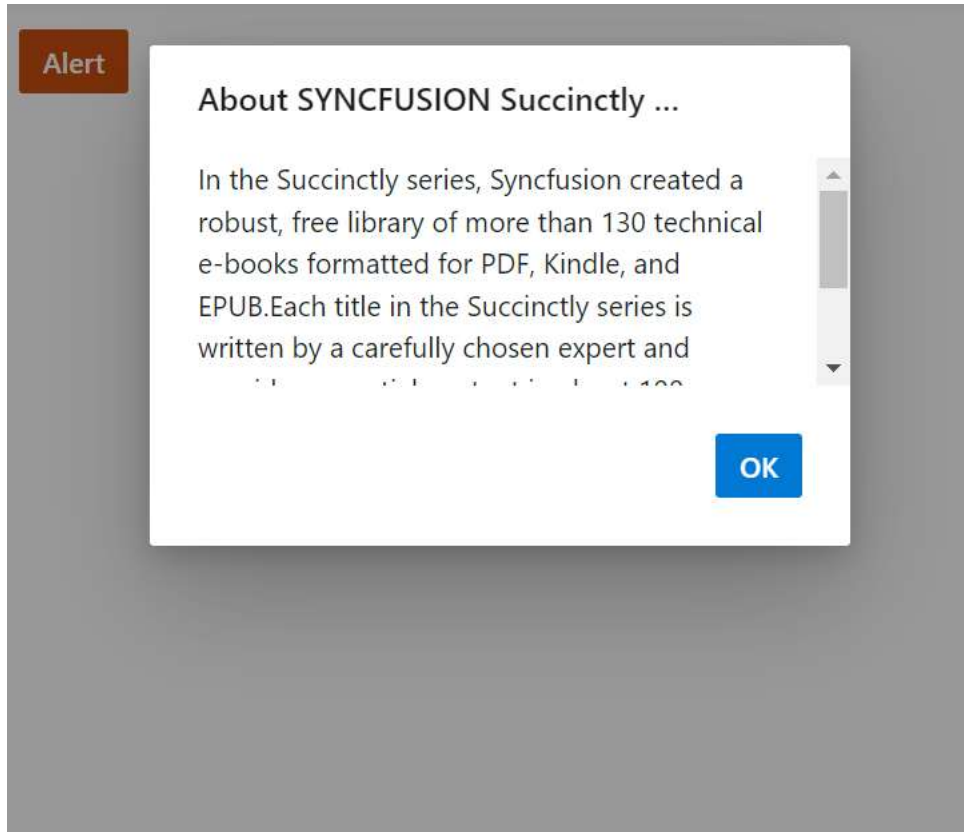
Use the following code to customize the min-width and min-height for alert dialog:

#### ALERT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Low battery",
 content: "10% of battery remaining",
 isDraggable: true
 });
 };
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}
```



## Customization of Predefined Dialogs

### Customize action buttons

You can customize the predefined dialogs buttons by using below properties.

- **okButton** - Use this property to customize **OK** button text and appearance.
- **cancelButton** - Use this property to customize **Cancel** button text and appearance.

Use the following code snippet for **Alert.cshtml**, **Confirm.cshtml** and **Prompt.cshtml** to customize the predefined dialog action buttons.

For alert dialog , customized the default dialog button content as **Done** by using the **okButton.text** property.

For confirm dialog, customized the default dialog buttons content as **Yes** and **No** by using the **okButton.text** and **cancelButton.text** property and also customized the dialog button icons by using **okButton.icon** property.

For prompt dialog , customized the default dialog buttons content as **Connect** and **Close** by using **okButton.text** and **cancelButton.text** property.

### Alert

#### ALERT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Good job!",
 content: "You clicked the alert button!",
 okButton: { text: 'Done' }
 });
 };
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Alert()
 {
 return View();
 }
}
```

### Confirm

#### CONFIRM.CSHTML

```
<div style="height:400px;">
```

```

 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
 success").Render()
</div>
<style>
 .e-btn-icon.e-icons.e-check icon.e-icon-left:before {
 content: '\e7ff';
 }
 .e-btn-icon.e-icons.e-close icon.e-icon-left:before {
 content: '\e7fc';
 }
</style>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.confirm({
 title: "Delete file",
 content: "Are you sure you want to permanently delete these
file?",
 okButton: { text: "Yes", icon: "e-icons e-check" },
 cancelButton: { text: "No", icon: "e-icons e-close" }
 });
 };
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}

```

### Prompt

#### PROMPT.CSHTML

```

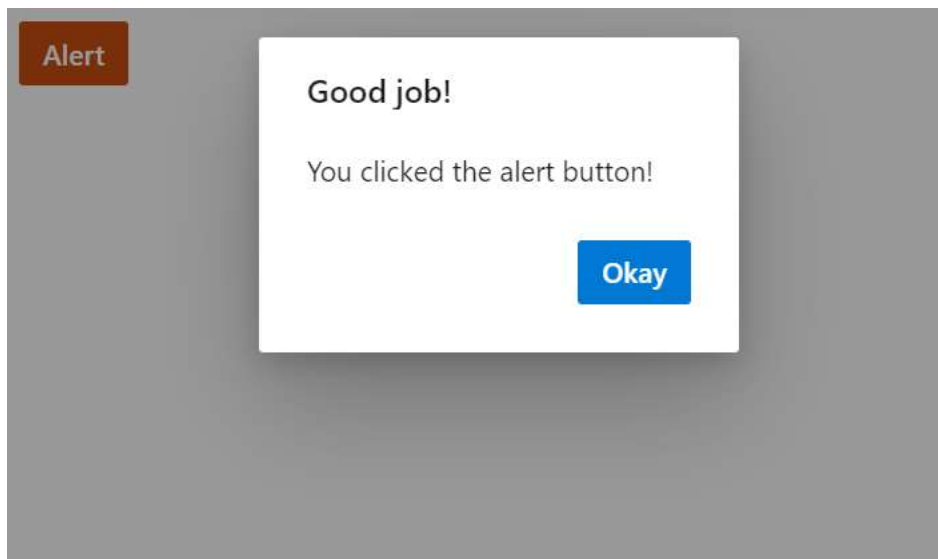
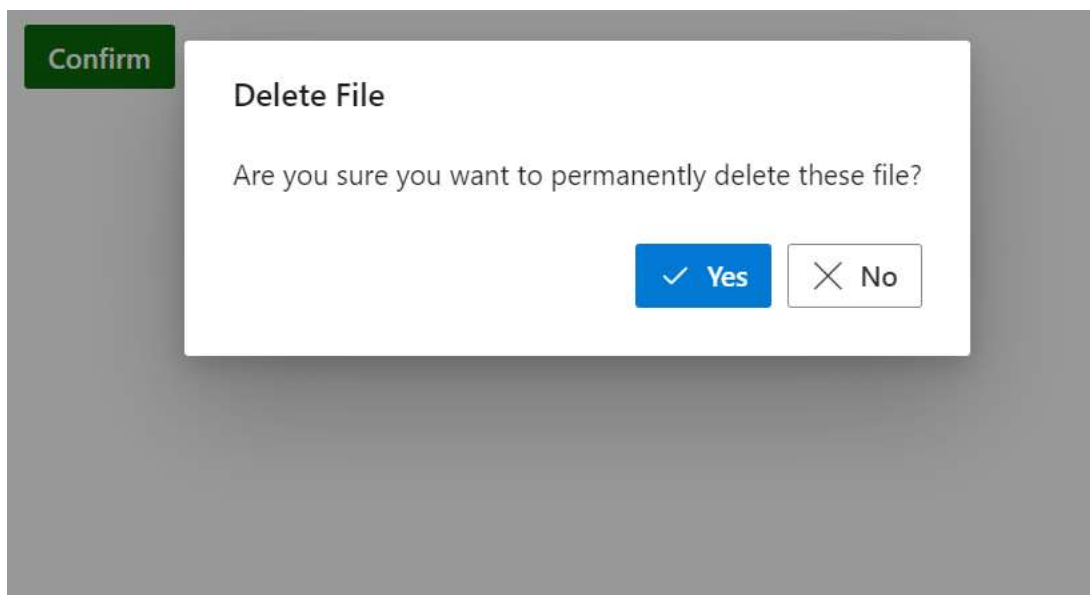
<div style="height:400px;">

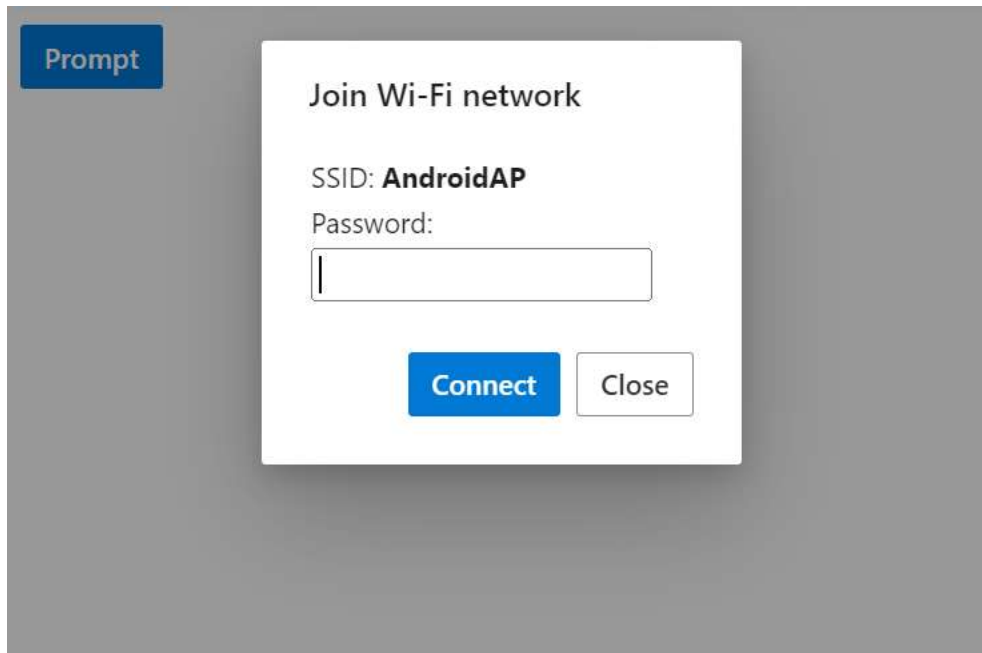
@Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()
</div>
<script>
 document.getElementById('promptBtn').onclick = function () {
 ej.popups.DialogUtility.confirm({
 title: "Join Wi-Fi network",
 content: "<table class=Table><tbody><tr><td>SSID:
AndroidAP</td></tr><tr> <td>Password:</td> </tr> <tr> <td> <input type=password id=password name=Required
class=e - input> </td> </tr> </tbody> </table>",
 okButton: { text: "Connect" },
 cancelButton: { text: "Close" }
 });
 };
</script>

```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}
```

**Results from the code snippet****Alert****Confirm****Prompt**



### Show or hide dialog close button

When rendering the predefined dialogs through utility methods, You can close the dialog using the following ways. The default values of `closeOnEscape` and `showCloseIcon` is `false`.

- By pressing the escape key if the [closeOnEscape](#) property is enabled.
- By clicking the close button if the [showCloseIcon](#) property is enabled.

You can also manually close the Dialogs by creating an instance to the dialog and call the [hide](#) method.

Use the following code for **alert**, **confirm** and **prompt** to demonstrates the different ways of hiding the utility dialog.

### Alert

#### ALERT.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("alertBtn").Content("Alert").CssClass("e-
danger").Render()
</div>
<script>
 document.getElementById('alertBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Low battery",
 content: "10% of battery remaining",
 showCloseIcon: true,
 closeOnEscape : true
 });
 };
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Alert()
 {
 return View();
 }
}
```

## Confirm

### CONFIRM.CSHTML

```
<div style="height:400px;">
 @Html.EJS().Button("confirmBtn").Content("Confirm").CssClass("e-
success").Render()
</div>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Delete multiple items",
 content: "Are you sure you want to permanently delete these
items?",
 showCloseIcon: true,
 closeOnEscape : true
 });
 };
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}
```

## Prompt

### PROMPT.CSHTML

```
<div style="height:400px;">

@Html.EJS().Button("promptBtn").Content("Prompt").IsPrimary(true).Render()
</div>
<script>
 document.getElementById('promptBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Join Wi-Fi network",
 content: "<table class=Table><tbody><tr><td>SSID:
AndroidAP</td></tr><tr> <td>Password:</td> </tr> <tr> <td> <input type=password id=password name=Required
class=e - input> </td> </tr> </tbody> </table>",
 showCloseIcon: true,
 closeOnEscape : true
 });
 };
</script>
```

```

 });
};
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Prompt()
 {
 return View();
 }
}

```

### Results from the code snippet

#### Alert

![ASP.NET MVC Predefined Dialogs alert close icon](../images/alert-show-hide-btn.png)

#### Confirm

![ASP.NET MVC Predefined Dialogs confirm close icon](../images/confirm-show-hide-btn.png)

#### Prompt

![ASP.NET MVC Predefined Dialogs prompt close icon](../images/prompt-show-hide-btn.png)

#### Customize dialog content

You can load custom content in predefined dialogs using the `content` property.

Use the following code to customize the dialog content to render the custom TextBox component inside the prompt dialog to get the username from the user.

### ALERT.CSHTML

```

<div style="height:400px;">
 @Html.EJS().Button("confirmBtn").Content("confirm").CssClass("e-
 success").Render()
</div>
<script>
 document.getElementById('confirmBtn').onclick = function () {
 ej.popups.DialogUtility.alert({
 title: "Join chat group",
 content: "<p>Enter your name: </p><input class=e-input />",
 width: "350px"
 });
 };
</script>

```

### CONTROLLER.CS

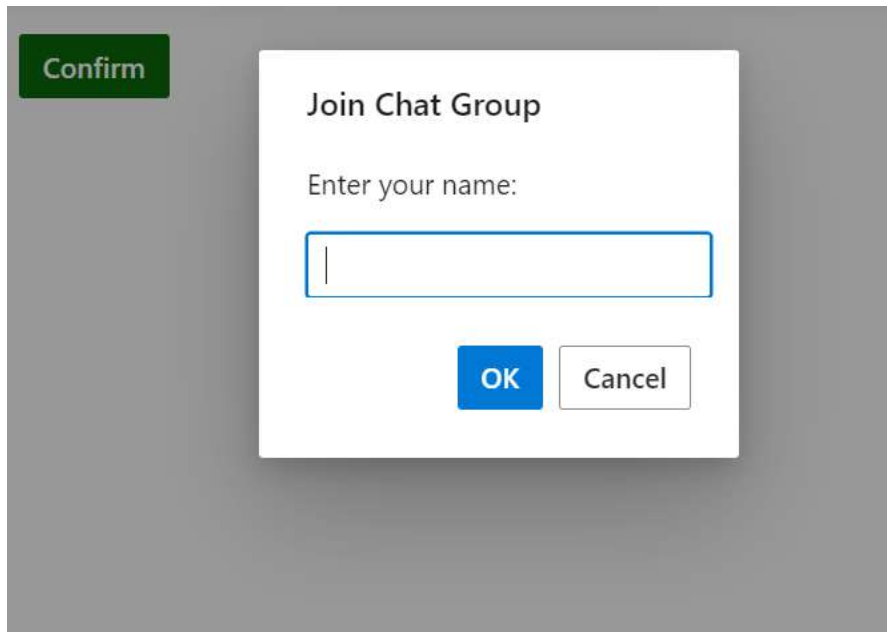
```

public class HomeController : Controller
{
 public ActionResult Confirm()
 {
 return View();
 }
}

```

```
}
}
```

### Results from the code snippet



## Progress Bar

### Getting Started with ASP.NET MVC Progress Bar Control

This section briefly explains about how to include [ASP.NET MVC Progress Bar](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5



NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

`

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

`

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

##### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

##### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

#### Add ASP.NET MVC Progress Bar control

Now, add the Syncfusion ASP.NET MVC Progress Bar control in **~/Views/Home/Index.cshtml** page.

##### CSHTML

```
@(Html.EJS().ProgressBar("container").Render())
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Progress Bar control will be rendered in the default web browser.

### Progress Type

You can change the type of progress bar by using [Type](#) property. By default **Linear** type of progress bar will render.

#### CSHTML

```
@(Html.EJS().ProgressBar("container")
 .Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(40)
 .Height("60")
 .Minimum(0).Maximum(100)
 .Render())
```



**Note:** [View Sample in GitHub.](#)

### Types

Visualize progress in different shapes (rectangle, circle, and semi-circle) to give a unique appearance to your app design.

#### Linear

<!-- markdownlint-disable MD033 -->

Set **Type** to Linear to get the linear progress bar. It also support secondary progress and different mode of progress.

#### CSHTML

```
@(Html.EJS().ProgressBar("lineardeterminate")
 .Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(100)
 .Height("60")
 .Minimum(0).Maximum(100)
 .Render())
@ (Html.EJS().ProgressBar("linearindeterminate")
 .Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(20)
 .Height("60").IsIndeterminate(true)
 .Minimum(0).Maximum(100)
 .Render())
@ (Html.EJS().ProgressBar("linearbuffer")
 .Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(40)
 .Height("60").SecondaryProgress(60)
 .Minimum(0).Maximum(100)
 .Render())
@ (Html.EJS().ProgressBar("linearsegment")
```

```
.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(100)
 .Height("60").SegmentCount(8)
 .Minimum(0).Maximum(100)
 .Render()
```

**LINEAR.CS****Circular**

Set **Type** to Circular to get the circular progress bar. It also support secondary progress and different mode of progress.

**CSHTML**

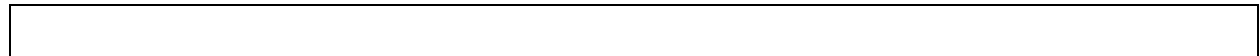
```
@(Html.EJS().ProgressBar("Circulareterminate")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(100)
 .Height("60")
 .Minimum(0).Maximum(100)
 .Render())
@(Html.EJS().ProgressBar("Circularindeterminate")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(20)
 .Height("60").IsIndeterminate(true)
 .Minimum(0).Maximum(100)
 .Render())
@(Html.EJS().ProgressBar("Circularbuffer")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(40)
 .Height("60").SecondaryProgress(60)
 .Minimum(0).Maximum(100)
 .Render())
@(Html.EJS().ProgressBar("Circularsegment")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(100)
 .Height("60").SegmentCount(8)
 .Minimum(0).Maximum(100)
 .Render())
```

**CIRCULAR.CS****Tooltip in ASP.NET MVC ProgressBar Component****Tooltip**

The tooltip for the progress bar is used to represent the progress value. During the initial load, it can be enabled by using the [Enable](#) property. The [ShowTooltipOnHover](#) property can show the tooltip on mouseover.

**CSHTML**

```
@(Html.EJS().ProgressBar("lineardeterminate")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(90)
 .Height("90")
 .Animation(an =>
an.Enable(true).Delay(0).Duration(2000))
 .Tooltip(tp => tp.Enable(true))
 .Render())
```

### **TOOLTIP.CS**

### Format

By default, the tooltip shows information about progress. In addition to that, show more information in the tooltip using the [Format](#) property.

### **CSHTML**

```
@(Html.EJS().ProgressBar("lineardeterminate")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(90)
 .Height("90")
 .Animation(an =>
an.Enable(true).Delay(0).Duration(2000))
 .Tooltip(tp => tp.Enable(true).Format("Progress:
${value}"))
 .Render())
```

### **FORMAT.CS**

### Customization

The [Fill](#) and [Border](#) properties are used to customize the background color and border of the tooltip respectively. The [TextStyle](#) property in the tooltip is used to customize the font of the tooltip text.

### **CSHTML**

```
@(Html.EJS().ProgressBar("lineardeterminate")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(90)
 .Height("90")
 .Animation(an =>
an.Enable(true).Delay(0).Duration(2000))
 .Tooltip(tp => tp.Enable(true).Format("Progress:
${value}").TextStyle(ts =>
ts.FontWeight("900").Size("15px").Color("red").FontFamily("Roboto").FontStyle("Italic")))
 .Render())
```

### **CUSTOMIZATION.CS**



## States

Visualize progress in different states.

### Determinate

<!-- markdownlint-disable MD033 -->

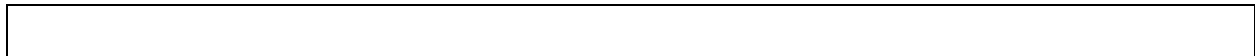
This is the default state. You can use it when the progress estimation is known.

#### CSHTML

```
@ (Html.EJS () .ProgressBar ("container")

.Type (Syncfusion.EJ2.ProgressBar.ProgressType.Linear) .Value (100)
 .Height ("60")
 .Minimum (0) .Maximum (100)
 .Render ())
```

#### DETERMINE.CS



### Indeterminate

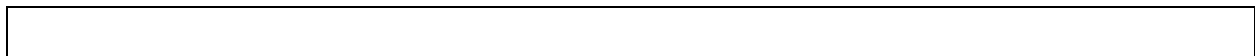
By enabling the **IsIndeterminate** property, the state of the progress bar can be changed to indeterminate when the progress cannot be estimated or is not being calculated. It can be combined with determinate mode to know that the application is estimating progress before the actual progress starts.

#### CSHTML

```
@ (Html.EJS () .ProgressBar ("container")

.Type (Syncfusion.EJ2.ProgressBar.ProgressType.Linear) .Value (20)
 .Height ("60") .IsIndeterminate (true)
 .Minimum (0) .Maximum (100)
 .Render ())
```

#### INDETERMINATE.CS



### Buffer

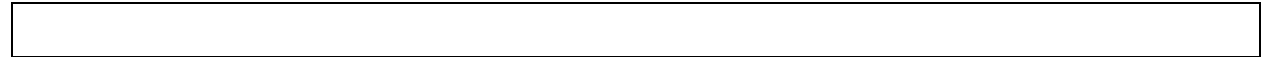
<!-- markdownlint-disable MD033 -->

You can use a secondary progress indicator when the primary progress depends on the secondary progress. This will allow users to visualize both primary and secondary progress simultaneously.

#### CSHTML

```
@ (Html.EJS () .ProgressBar ("container")
```

```
.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(40)
 .Height("60").SecondaryProgress(60)
 .Minimum(0).Maximum(100)
 .Render()
```

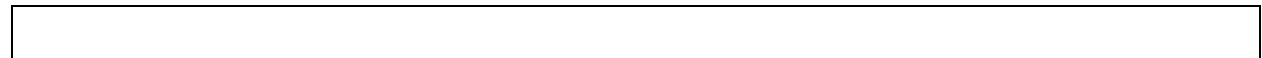
**BUFFER.CS****Range**

<!-- markdownlint-disable MD033 -->

Range represents the entire span of the progress bar and can be defined using the **Minimum** and **Maximum** properties. The default value of the range is 0 to 100.

**CSHTML**

```
@(Html.EJS().ProgressBar("container")
 .Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(40)
 .Height("60")
 .Minimum(0).Maximum(100)
 .Render())
```

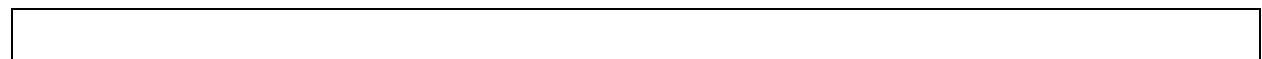
**RANGE.CS****Customization****Segments**

<!-- markdownlint-disable MD033 -->

We can divide a progress bar into multiple segments using a **SegmentCount** to visualize the progress of multiple sequential tasks.

**CSHTML**

```
@(Html.EJS().ProgressBar("container")
 .Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(100)
 .Height("60").SegmentCount(8)
 .Minimum(0).Maximum(100)
 .Render())
```

**SEGMENTS.CS**

### Thickness

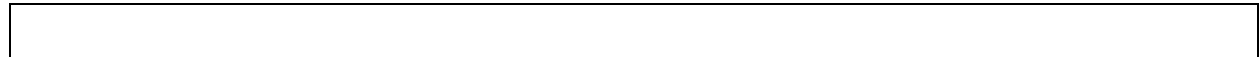
Customize the thickness of the track using [TrackThickness](#), progress using [ProgressThickness](#) and secondary progress using [SecondaryProgressThickness](#) to render the progress bar with different appearances.

#### CSHTML

```
@(Html.EJS().ProgressBar("container").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear)
 .Value(100).ShowProgressValue(true)

.Width("90%).Height("60").TrackThickness(24).ProgressThickness(24).Minimum(
0).Maximum(100).SecondaryProgressThickness(20)
 .LabelStyle(ls => ls.Color("#FFFFFF"))
 .Animation(an => an.Enable(true).Delay(0).Duration(2000))
 .Render())
```

#### THICKNESS.CS



### Radius

<!-- markdownlint-disable MD033 -->

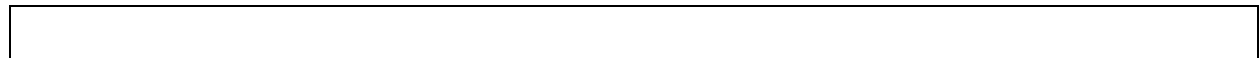
The radius of the progress bar can be customized using **Radius** property and corner can be customized by **CornerRadius** property.

#### CSHTML

```
@(Html.EJS().ProgressBar("container").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(60)
 .TrackThickness(80).ProgressThickness(10).Radius("100%")

.ProgressColor("white").CornerRadius(Syncfusion.EJ2.ProgressBar.CornerType.Round)
 .Minimum(0).Maximum(100)
 .Render())
```

#### RADIUS.CS



### InnerRadius

<!-- markdownlint-disable MD033 -->

The inner radius of the progress bar can be customized using **InnerRadius** property.

#### CSHTML

```
@(Html.EJS().ProgressBar("container").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(60)
 .Width("160px").Height("160px").EnableRtl(false)
```

```
.TrackThickness(80).ProgressThickness(10).Radius("100%").InnerRadius("190%")
.ProgressColor("white").CornerRadius(Syncfusion.EJ2.ProgressBar.CornerType.Round)
.Minimum(0).Maximum(100)
.Render()
```

### **INNER-RADIUS.CS**

#### Progress color and track color

<!-- markdownlint-disable MD033 -->

Customize the color of progress, secondary progress, and track by using the [ProgressColor](#), [SecondaryProgressColor](#) and [TrackColor](#) properties.

### **CSHTML**

```
@(Html.EJS().ProgressBar("percentage").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear)
 .Value(100).ShowProgressValue(true).TrackColor("#F8C7D8")
 .Width("90%").Height("60").CornerRadius(Syncfusion.EJ2.ProgressBar.CornerType.Round)
 .ProgressColor("#E3165B").TrackThickness(24).ProgressThickness(24).Minimum(0)
 .Maximum(100).SecondaryProgress(60).SecondaryProgressColor("green")
 .Animation(an => an.Enable(true).Delay(0).Duration(2000))
 .LabelStyle(ls => ls.Color("#FFFFFF"))
 .TextRender("textRender")
 .Render())
var textRender = function (args) {
 args.text = '50';
}
```

### **TRACK-THICKNESS.CS**

#### Annotation and Label

##### Annotation

In the circular progress bar, you can add any view to the center using the **Content** property in annotation.

For example, you can include add, start, or pause button to control the progress. You can also add an image that indicates the actual task in progress or add custom text that conveys how far the task is completed.

### **CSHTML**



```
@(Html.EJS().ProgressBar("container").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(60)

.TrackThickness(80).InnerRadius("190%").CornerRadius(Syncfusion.EJ2.ProgressBar.CornerType.Round)
 .Minimum(0).Maximum(100).Animation(an =>
an.Enable(true).Delay(0).Duration(2000))
 .Annotations(an => { an.Content("<div style='font-size:20px; font-weight:bold; color:#ffffff;fill:#ffffff'>60%</div>").Add();
}).Render())
```

### ANNOTATION.CS

### Label

You can show the progress value in both linear and circular progress bar using **ShowProgressValue** property.

### CSHTML

```
@(Html.EJS().ProgressBar("container").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(50)

.ShowProgressValue(true).TrackThickness(24).ProgressThickness(24)
 .Minimum(0).Maximum(100)
 .Render())
```

### LABEL.CS

### Animation

<!-- markdownlint-disable MD033 -->

Progress Bar support to animate the progress by using **Animation** property. Enable the animation by setting **Enable** property and also you can control the speed by using **Duration** property.

### CSHTML

```
@(Html.EJS().ProgressBar("container")

.Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(40)
 .Height("60")
 .Minimum(0).Maximum(100)
 .Animation(an =>
an.Enable(true).Delay(0).Duration(2000))
 .Render())
```

### ANIMATION.CS



## Events

### ValueChanged

<!-- markdownlint-disable MD033 -->

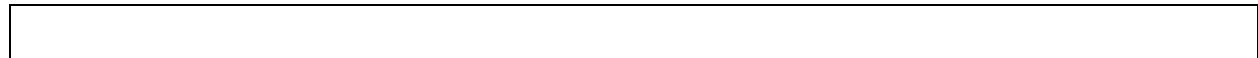
This event is triggered when the progress value is changed. This event contains the following event argument.

#### CSHTML

```
@(Html.EJS().ProgressBar("lineardeterminate").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Linear).Value(100)
 .Height("60").Load("Load")

.Minimum(0).Maximum(100).ValueChanged("progressValue")
 .Animation(an =>
an.Enable(true).Delay(0).Duration(2000))
 .Render())
```

#### VALUE-CHANGED.CS



### ProgressCompleted

This event is triggered when the progress attains the Maximum value. This event contains the following argument.

#### CSHTML

```
<div id="pause-container">
 @(Html.EJS().ProgressBar("pause-
container").Type(Syncfusion.EJ2.ProgressBar.ProgressType.Circular).Value(100)
)
 .Width("160px").Height("160px")
 .Minimum(0).Maximum(100).Load("progressLoad")
 .Animation(an =>
an.Enable(true).Delay(0).Duration(2000)).ProgressCompleted("progressComplete
d")
 .Annotations(an =>
 {
 an.Content("<div style='font-size:20px; font-weight:bold;
color:#ffffff;fill:#ffffff'>60%</div>").Add();
 }).Render())
</div>
<script>
var clearTimeout1;
var clearTimeout2;
var progressCompleted = function (args) {
 clearTimeout(clearTimeout2);
 clearTimeout2 = setTimeout(function () {
 var pausePlay = document.getElementById("pause-
container").ej2_instances[0];
```

```

 pausePlay.annotations[0].content = "<div id='point1'
style='font-size:20px;font-
weight:bold;color:#ffffff;fill:#ffffff'>60%</div>";
 pausePlay.dataBind();
 }, 2000);
}
</script>

```

### **PROGRESS-COMPLETED.CS**

### Accessibility in ASP.NET MVC Progress bar component

The Progress bar component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Progress bar component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

```

}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

The Progress bar component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Progress bar component:

- progressbar (role)
- aria-valuemin (attribute)
- aria-valuemax (attribute)
- aria-valuenow (attribute)
- aria-label (attribute)

### Keyboard interaction

The Progress bar component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Progress bar component.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves the focus to the Progress bar element. |

| **Ctrl + P** | Prints the Progress bar. |

### Ensuring accessibility

The Progress bar component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Progress bar component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Progress bar component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## Progress Button

### Getting Started with ASP.NET MVC Progress Button Control

This section briefly explains about how to include [ASP.NET MVC Progress Button](#) control in your ASP.NET MVC application using Visual Studio.

## Prerequisites

### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

## Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

## Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

## Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

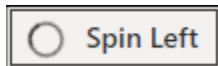
Add ASP.NET MVC Progress Button control

Now, add the Syncfusion ASP.NET MVC Progress Button control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().ProgressButton("element").Content("Spin Left").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Progress Button control will be rendered in the default web browser.



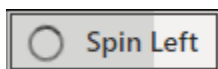
**Note:** Progress Button supports different styles, types and sizes like [Button](#). In addition, it also support `top` and `bottom` position of the icon.

Enabling progress in button

You can enable the background filler UI by setting the [EnableProgress](#) property to `true`.

#### CSHTML

```
@Html.EJS().ProgressButton("spinleft").Content("Spin
Left").EnableProgress(true).Render()
```



**Note:** [View Sample in GitHub](#).

See also

- [Spinner and Progress options](#)

<!-- markdownlint-disable MD002 MD022 -->

## Spinner

### *Change spinner position*

Spinner position can be changed by modifying the [Position](#) property of [SpinSettings](#). By default, the spinner is positioned at the left of the ProgressButton. You can position it at the **left**, **right**, **top**, **bottom**, or **center** of the text content.

### *Change spinner size*

Spinner size can be changed by modifying the [Width](#) property of [SpinSettings](#). In this demo, the **Width** is set to **20** to change the spinner size.

### *Spinner template*

You can use custom spinner by specifying the [Template](#) property of [SpinSettings](#) with custom styles.

The following sample demonstrates the above functionalities of the spinner.

## **CSHTML**

```
@Html.EJS().ProgressButton("submit").Content("Submit").SpinSettings(ViewBag.spinSettings).Render()
<style>
@@keyframes custom-rolling {
 0% {
 -webkit-transform: rotate(0deg);
 transform: rotate(0deg);
 }
 100% {
 -webkit-transform: rotate(360deg);
 transform: rotate(360deg);
 }
}
.template {
 border: 2px solid green;
 border-style: dotted;
 border-radius: 50%;
 border-top-color: transparent;
 border-bottom-color: transparent;
 height: 16px;
 width: 16px;
}
.template {
 -webkit-animation: custom-rolling 1.3s linear infinite;
 animation: custom-rolling 1.3s linear infinite;
}
</style>
```

## **DEFAULT.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ProgressButtonSpinSettings spinSettings = new
 ProgressButtonSpinSettings()
 {
 Position = SpinPosition.Right,
 Width = "20",
 }
 }
}
```

```

 Template = "<div class='template'></div>"
 };
 ViewBag.spinSettings = spinSettings;
 return View();
}

```

<!-- markdownlint-disable MD025 MD022 -->

## Progress

### Content animation

The [Content](#) of the ProgressButton can be animated during progress using the [Effect](#) property of [AnimationSettings](#). You can also set custom duration and timing function using the [duration](#) and [easing](#) properties. The possible [effect](#) values are [None](#), [SlideLeft](#), [SlideRight](#), [SlideUp](#), [SlideDown](#), [ZoomIn](#), and [ZoomOut](#).

### CSHTML

```

@Html.EJS().ProgressButton("slideleft").Content("Slide
Left").EnableProgress(true).AnimationSettings(ViewBag.animationSettings).SpinSettings(ViewBag.spinSettings).Render()

```

### DEFAULT.CS

```

using Syncfusion.EJ2.SplitButtons;
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ProgressButtonSpinSettings spinSettings = new
ProgressButtonSpinSettings()
 {
 Position = SpinPosition.Left
 };
 ViewBag.spinSettings = spinSettings;
 ProgressButtonAnimationSettings animationSettings = new
ProgressButtonAnimationSettings()
 {
 Effect = AnimationEffect.SlideRight,
 Duration = 500,
 Easing = "linear",
 };
 ViewBag.animationSettings = animationSettings;
 return View();
 }
}

```

### Change step of the ProgressButton

The progress state can be visualized to the specified interval by changing the [Step](#) property in the [begin](#) event of the ProgressButton. In this demo, the [Step](#) property is set to [20](#) to show progress at every 20% increment.

### CSHTML



```
@Html.EJS().ProgressButton("progress").Content("Progress
Step").EnableProgress(true).Begin("begin").CssClass("e-hide-
spinner").Render()
<script>
 function begin(args) {
 args.step = 20;
 }
</script>
```

**DEFAULT.CS**

```
public ActionResult Default()
{
 return View();
}
```

**Note:** The class `e-hide-spinner` hides the spinner in the ProgressButton, For more information, see [hide spinner](#) section

*Changing progress state dynamically*

The progress state can be changed dynamically by modifying the `Percent` property in the progress events. In this demo, on 40% completion of progress, the `Percent` property is set to `90` to show dynamic change of the progress state.

**CSHTML**

```
@Html.EJS().ProgressButton("progress").Content("Progress").EnableProgress(tr
ue).Duration(15000).Begin("begin").Progress("progress").End("end").CssClass(
"e-hide-spinner").Render()
<script>
 function begin(args) {
 this.content = 'Progress ' + args.percent + '%';
 this.dataBind();
 }
 function progress(args) {
 this.content = 'Progress ' + args.percent + '%';
 this.dataBind();
 if (args.percent === 40) {
 args.percent = 90
 }
 }
 function end(args) {
 this.content = 'Progress ' + args.percent + '%';
 this.dataBind();
 }
</script>
```

**DEFAULT.CS**

```
public ActionResult Default()
{
 return View();
}
```

**Note:** The method `dataBind` applies the property changes immediately to the component.

## Start and stop methods

You can pause and resume the progress using the `stop` and `start` methods, respectively. In this demo, clicking the `ProgressButton` will pause and resume the progress.

**CSHTML**

```
@Html.EJS().ProgressButton("download").Content("Download").EnableProgress(true).IconCss("e-btn-sb-icon e-download").Duration(4000).Created("created").End("end").CssClass("e-hide-spinner").Render()
<script>
 function end() {
 var progressBtn =
ej.base.getComponent(document.querySelector("#download"), "progress-btn");
 progressBtn.content = 'Download';
 progressBtn.iconCss = 'e-btn-sb-icon e-download';
 progressBtn.dataBind();
 }
 function created() {
 var progressBtn =
ej.base.getComponent(document.querySelector("#download"), "progress-btn");
 progressBtn.element.addEventListener('click', clickHandler);
 }
 function clickHandler() {
 var progressBtn =
ej.base.getComponent(document.querySelector("#download"), "progress-btn");
 if (progressBtn.content === 'Download') {
 progressBtn.content = 'Pause';
 progressBtn.iconCss = 'e-btn-sb-icon e-pause';
 progressBtn.dataBind();
 }
 // clicking on ProgressButton will stop the progress when the text
 content is 'Pause'
 else if (progressBtn.content === 'Pause') {
 progressBtn.content = 'Resume';
 progressBtn.iconCss = 'e-btn-sb-icon e-play';
 progressBtn.dataBind();
 progressBtn.stop();
 }
 // clicking on ProgressButton will start the progress when the text
 content is 'Resume'
 else if (progressBtn.content === 'Resume') {
 progressBtn.content = 'Pause';
 progressBtn.iconCss = 'e-btn-sb-icon e-pause';
 progressBtn.dataBind();
 progressBtn.start();
 }
 }
}
</script>
<style>
 @@font-face {
 font-family: 'btn-icon';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfgAAAEoAAAAVmNtYXDNH+dzAAABoAAAAEJnbHlm1v4
8pAAAAfgAAQYAGVhZBOPfZcAAADQAAAAANmhoZWEIUQQJAAAArAAAACRobXR4IAAAAAAAAAAAAAA
```

```

gbG9jYQN6ApQAAAHkAAAAEm1heHABFQCqAAABCAAAACBuYw1l071fXAAABhAAAAIxcG9zdK9uovo
AAAhEAAAAgAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAACAABAAAAAQAAJ1LUzF8
PPPUACwQAAAAAANG+nFMAAAAA2D6cUwAAAAAD9AP0AAAACAACAAAAAAAAAAAAAEAAAAIAJ4AAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnBgQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAAAACAAAAAwAAABQAAwABAAAAFAAEAC4AAAAEAAQAAQA
A5wb//wAA5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcAAAAAAAAAADgAkADIAhAEuAewCDAAAAAE
AAAAAA2ED9AACAAA3CQGeAsT9PAwB9AH0AAACAAAAAAPHA/QAAwAHAAALIREhASERIQJpAV7+ov3
QAV7+ogwD6PwYA+gAAAEAAAAAA4sD9AACAAATARF0AxgCAP4MA+gAAAABAAAAAAP0A/QAQwAAExE
fDyE/DxEvDyEPDgWBAgMFBQcICQkLCwwMDQ4NAtONDg0MDAsLCQkIBwUFAwIBAQIDBQUHCAkJCws
MDA00Df0mDQ4NDawLCwkJCAcFBQMCA239Jg4NDQ0LCwsJCQgHBQUdAgEBAGMFBQcICQkLCwsNDQ0
OAtO0DQ0NCwsLCQkIBwUFAwIBAQIDBQUHCAkJCwsLDQ0NAAIAAAAAA/MDxQADAIwAADczESMBDwM
VFw8METM3HwQ3Fz8KPQEVBt8LLwg3NT8INS8FNT8NNS8JBjYU/BDUvCyMPAQytrQH5AgoEAQEBArg
hERESEyIJCsgQBIEHNQceOZPbDgUICw0LCQUDBAICBAkGAgEBAQMOBAkIBgcDAwEBAQEDAwMJAgE
BAxYLBQQEAwMCAGIEBAoBAQEECgcHBgUFBAMDAQEBAQQFBwkFBQUGEf6tDwKEAwIBAQMDCgwVAwc
GDAsNBwdaAYcB3gEFAwN2HwoELDodGxwaLwkIGwz+igEBHwMBAQECAQEDBgoKDAYICAgFCakICwU
EBAQFAwYDBwgIDAgHCACGBgYFBQkEAgYCBawJBgUGBwkJCgkICAcLBAIFAwIEBAQFBQcGBwgHBgY
GBgoJCAYCAGeBAQFGMRkaGw0NDA0LIh4xBQcBAEBAGADAAAAAOKA/MAHABCAJ0AAAEzHwIRDwM
hLwIDNZM/CjUTHwcVIwcVIy8HETcXMz8KNScxBxEfDjSBHQEFDTMhMz8OES8PIz0BLw4hA0EDBQQ
DAQIEBf5eBQQCAW4RDg0LCQgGBQUDBAFEBAMDawIBAQGL7Y0EAWQCAgIBAYYKChEQDQsJCACeBAU
CYt8BAQIDBAUFBQcHBwgICQgKjQECAGMEBAUFBgYHBgcIBwGcCAcHBWYGBgUFBAQDAGIBAQEBAgI
DBAQFBQYGBgcHBwgmAQMDAwUFBgYHBwgICQkJ/tQCiwMEBf3XAwYEAgIEBgFoAQEDBQYGBwgIBw0
KhQEiAQEBAGMDAwTV+94BAQECawMDBAGyAQECBAYHCAGJCgkQCACQ6/47CQkICQcIBwYGBQQEAwI
CUAGHBwcGBgYFBQQEAwMBAgIBAwMEBAUFBQcGBwCHCAImCAcHBWYGBgUFBAQDAGIBADUJCQgICAg
GBwYFBAQDAGEBAAAAAIAAAAAA6cD9AADAAwAADchNSElAQcJAScBESNZAO78sgGB/uMuAXkBgDb
+1EwMTZcBCD3+ngFiPf7pAxMAAAAAABIA3gABAAAAAIAAAAAAABAAAAAABAAgAAQABAAAAAA
CAACACQABAAAAAADAAgAEABAAAAAIAAAAgAGAABAAAAAFAAAsAIAABAAAAAAGAAgAKwABAAA
AAAAKACwAMwABAAAAAALABIAxwADAAEECQAAAAIAcQADAAEECQABABAAcWADAAEECQACAA4AgwA
DAAEECQADABAAkQADAAEECQAEABAAoQADAAEECQAFABYAsQADAAEECQAGABAAxwADAAEECQAKAFg
AlwADAAEECQALACQBLyBidG4taWNvblJlZ3VsYXJidG4taWNvbmlJ0b1lpY29uVmVyc2lvbiAxLjB
idG4taWNvbmkzbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN
5bmNmdXNpb24uY29tACAAYgB0AG4ALQBpAGMAbwBuAFIAZQBnAHUAbABhAHIAIAYgB0AG4ALQBpAGM
AbwBuAGIAdABuAC0AaQBJAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYgB0AG4ALQBpAGMAbwB
uAEYAbwBuAHQAIAbnAGUAbgBlAHIAIAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHM
AaQBvAG4AIAbnAGUAdABYAG8AIAbTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQB
vAG4ALgBjAG8AbQAAAAACAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAAIAAAAAA
GAQcBCAEJAaptZWRpYS1wbGF5c21lZGlhLXBhdXNlDmFycm93aGVhZC1sZWZ0BHN0b3AJbGlrc2S0
tLTAXBGnVcHkQLWRvd25sb2FkLTAyLXdmLQAQ format('truetype');
font-weight: normal;
font-style: normal;
}
.e-btn-sb-icon {
font-family: 'btn-icon' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-download::before {
content: '\e706';
}
.e-play::before {

```

```

 content: '\e700';
 }
 .e-pause::before {
 content: '\e701';
 }
}
</style>

```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

### See Also

- [How to hide spinner](#)
- [Customize ProgressButton using cssClass](#)

### Accessibility in Progress Button Component

The Progress button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Progress button component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

```

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

The Progress button component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Progress button component:

| Attributes | Purpose |

| --- | --- |

| **aria-label** | Provides an accessible name for the icon only Progress button. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

### Keyboard interaction

The Progress button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Progress button component.

| **Press** | **To do this** |

| --- | --- |

| **Enter / Space** | Starts the progress. |

### Ensuring accessibility

The Progress button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Progress button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Progress button component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

## Styles and Appearances

To modify the ProgressButton appearance, you need to override the default CSS of ProgressButton component. Find the list of CSS classes and its corresponding section in ProgressButton. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

### CSS Class | Purpose of Class

- |.e-progress-btn|To customize the progress button
- |.e-progress-btn: hover|To customize the progress button on hover
- |.e-progress-btn: focus|To customize the progress button on focus
- |.e-progress-btn .e-spinner-pane .e-spinner-inner svg .e-path-circle|To customize the progress button spinner

## How To

### Change the text content and styles of the ProgressButton during progress

You can change the text content and styles of the ProgressButton during progress state by changing the text content and the [cssClass](#) property at the [begin](#) and [end](#) events.

### CSHTML

```
@Html.EJS().ProgressButton("upload").Content("Upload").EnableProgress(true).
CssClass("e-hide-spinner").Duration(4000).Begin("begin").End("end").Render()
<script>
 function begin() {
 this.content = 'Uploading...';
 this.cssClass = 'e-hide-spinner e-info';
 this.dataBind();
 }
 function end() {
 this.content = 'Success';
 this.cssClass = 'e-hide-spinner e-success';
 this.dataBind();
 setTimeout(() => {
 this.content = 'Upload';
 this.cssClass = 'e-hide-spinner';
 this.dataBind();
 }, 500)
 }
</script>
```

### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

### Customize progress using cssClass

You can customize the background filler UI using the [cssClass](#) property.

- Adding `e-vertical` to `cssClass` shows vertical progress.

- Adding `e-progress-top` to `cssClass` shows progress at the top.

You can also show reverse progress by adding custom class to the [cssClass](#) property.

#### CSHTML

```
@Html.EJS().ProgressButton("vprogress").Content("Vertical
Progress").EnableProgress(true).CssClass("e-hide-spinner e-
vertical").Duration(4000).Render()
@Html.EJS().ProgressButton("tprogress").Content("Progress
Top").EnableProgress(true).CssClass("e-hide-spinner e-progress-
top").Duration(4000).Render()
@Html.EJS().ProgressButton("rprogress").Content("Reverse
Progress").EnableProgress(true).CssClass("e-hide-spinner e-reverse-
progress").Duration(4000).Render()
<style>
 .e-reverse-progress .e-progress {
 right: 0;
 left: auto;
 }
 button {
 margin: 25px;
 }
</style>
```

#### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

#### Hide spinner

You can hide spinner in the ProgressButton by setting the `e-hide-spinner` property to [cssClass](#).

#### CSHTML

```
@Html.EJS().ProgressButton("progress").Content("Progress").EnableProgress(tr
ue).CssClass("e-hide-spinner").Render()
```

#### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

## QueryBuilder

### Getting Started with ASP.NET MVC Query Builder Control

This section briefly explains about how to include [ASP.NET MVC Query Builder](#) control in your ASP.NET MVC application using Visual Studio.

## Prerequisites

### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

## Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

## Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

## Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### **~/ \_LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```



**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Query Builder control

Now, add the Syncfusion ASP.NET MVC Query Builder control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@model List<QueryBuilderSample.Controllers.EmployeeView>
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(Model).Render()
```

#### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(EmployeeView.GetAllRecords());
 }
}

public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string LastName,
string Title, DateTime BirthDate, DateTime HireDate, int ReportsTo, string
Address, string PostalCode, string Phone, string City, string Country)
 {
 this.EmployeeID = EmployeeID;
 }
}
```

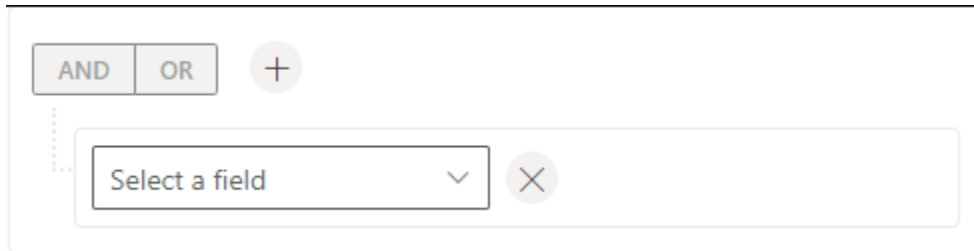
```

this.FirstName = FirstName;
this.LastName = LastName;
this.Title = Title;
this.BirthDate = BirthDate;
this.HireDate = HireDate;
this.ReportsTo = ReportsTo;
this.Address = Address;
this.PostalCode = PostalCode;
this.Phone = Phone;
this.City = City;
this.Country = Country;
}

public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public string Title { get; set; }
public DateTime BirthDate { get; set; }
public DateTime HireDate { get; set; }
public int ReportsTo { get; set; }
public string Address { get; set; }
public string PostalCode { get; set; }
public string Phone { get; set; }
public string City { get; set; }
public string Country { get; set; }
public static List<EmployeeView> GetAllRecords()
{
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales Representative", new
 DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2, "507 - 20th Ave.
 E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ", "USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President, Sales", new
 DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W. Capital Way",
 "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales Representative",
 new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3, " 4110 Old
 Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales Representative",
 new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6, "14 Garrett Hill
 ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales Manager", new
 DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8, "Coventry House Miner
 Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales Representative", new
 DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2, " 7 Houndstooth Rd.",
 " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales Representative", new
 DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7, "Edgeham
 Hollow Winchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "UK"));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales Coordinator",
 new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9, "722 Moss Bay
 Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales Representative", new
 DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5, "4726 - 11th Ave.
 N.E. ", "98105 ", "(71) 555-5598 ", "London", "UK "));
 return Emp;
}
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Query Builder control will be rendered in the default web browser.



### Render with rule

The following example shows a basic Query Builder component with rule.

#### CSHTML

```
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(Model).Rule(ViewBag.rule).Render()
```

#### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title", Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 return View(EmployeeView.GetAllRecords());
 }
}

public class EmployeeView
{
 public EmployeeView()
 {

```

```

}
public EmployeeView(int EmployeeID, string FirstName, string LastName,
string Title, DateTime BirthDate, DateTime HireDate, int ReportsTo, string
Address, string PostalCode, string Phone, string City, string Country)
{
this.EmployeeID = EmployeeID;
this.FirstName = FirstName;
this.LastName = LastName;
this.Title = Title;
this.BirthDate = BirthDate;
this.HireDate = HireDate;
this.ReportsTo = ReportsTo;
this.Address = Address;
this.PostalCode = PostalCode;
this.Phone = Phone;
this.City = City;
this.Country = Country;
}
public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public string Title { get; set; }
public DateTime BirthDate { get; set; }
public DateTime HireDate { get; set; }
public int ReportsTo { get; set; }
public string Address { get; set; }
public string PostalCode { get; set; }
public string Phone { get; set; }
public string City { get; set; }
public string Country { get; set; }
public static List<EmployeeView> GetAllRecords()
{
List<EmployeeView> Emp = new List<EmployeeView>();
Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales Representative", new
DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2, "507 - 20th Ave.
E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ", "USA"));
Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President, Sales", new
DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W. Capital Way",
"98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales Representative",
new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3, " 4110 Old
Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales Representative",
new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6, "14 Garrett Hill
", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales Manager", new
DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8, "Coventry House Miner
Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", " USA"));
Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales Representative", new
DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2, " 7 Houndstooth Rd.",
" WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
Emp.Add(new EmployeeView(7, "Robert", "King", "Sales Representative", new
DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7, "Edgeham
Hollow Winchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", " UK"));
Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales Coordinator",
new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9, "722 Moss Bay
Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
}

```

```
Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales Representative", new
DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5, "4726 - 11th Ave.
N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
return Emp;
}
}
```

The screenshot shows a web-based Query Builder interface. At the top, there are buttons for 'AND', 'OR', and a '+' icon to add more conditions. Below these, there are two filter rules displayed in a list. The first rule has the column 'Employee ID', the operator 'Equal', and the value '1'. The second rule has the column 'Title', the operator 'Equal', and the value 'Sales Manager'. Each rule has a close button (X) to its right. On the left side of the filter rules, there is a vertical list of column names: Employee ID, Title, Hire Date, Salary, Commission Pct, Manager, Department Name, Location, and Country.

**Note:** [View Sample in GitHub.](#)

### Column Binding

The column definitions are used as the [DataSource](#) schema in the Query Builder. This plays a vital role in rendering column values. The query builder operations such as create or delete conditions and create or delete groups are performed based on the column definitions. The [Field](#) property of the [Columns](#) is necessary to map the data source values in the query builder columns.

**Note:** If the column field is not specified in the data source, the column values will be empty.

### Auto generation

The [Columns](#) are automatically generated when the [Columns](#) declaration is empty or undefined while initializing the query builder. All the columns in the [DataSource](#) are bound as the query builder columns.

### CSHTML

```
@Html.EJS().QueryBuilder("querybuilder").Width("100%").DataSource(ViewBag.da
taSource).Render()
```

### DEFAULT.CS

```
public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule{ Label="Title", Field="Title", Type="string",
Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}
```

```

public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }

 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry HouseMiner Rd. ", "EC2 7JR ", " (206) 555-8122", "Tacoma ", "
USA"));
 }
}

```

```

 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,
"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK"));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
 }
}

```

**Note:** When columns are auto-generated, the column type will be determined from the first record of the data source.

### Labels

By default, the column label is displayed from the column [Field](#) value. To override the default label, you have to define the [Label](#) value.

**Note:** If both the field and headerText are not defined in the column, the column renders with “empty” header text.

### Operators

The operator for a column can be defined in the columns property.

The available operators and its supported data types are:

| Operators          | Description                                                             | Supported Types              |
|--------------------|-------------------------------------------------------------------------|------------------------------|
| -----              | -----                                                                   | -----                        |
| startswith         | Checks whether the value begins with the specified value.               | String                       |
| endswith           | Checks whether the value ends with the specified value.                 | String                       |
| contains           | Checks whether the value contains the specified value.                  | String                       |
| equal              | Checks whether the value is equal to the specified value.               | String Number/Date/Boolean   |
| notequal           | Checks for values not equal to the specified value.                     | String/Number  Date  Boolean |
| greaterthan        | Checks whether the value is greater than the specified value.           | Date/Number                  |
| greaterthanorequal | Checks whether a value is greater than or equal to the specified value. | Date/Number                  |
| lessthan           | Checks whether the value is less than the specified value.              | Date/Number                  |
| lessthanorequal    | Checks whether the value is less than or equal to the specified value.  | Date/Number                  |
| between            | Checks whether the value is between the two-specific value.             | Date/Number                  |
| notbetween         | Checks whether the value is not between the two-specific value.         | Date/Number                  |

| in | Checks whether the value is one of the specific values. | String/Number |

| notin | Checks whether the value is not in the specific values. | String/Number |

### Step

The Query Builder allows you to set the step values to the number fields. So that, you can easily access the numeric textbox. Use the [Step](#) property, to set the step value for number values.

### Format

The Query Builder formats date and number values. Use the [Format](#) property, to format date and number values.

### CSHTML

```
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee
ID").Type("number").Step(10).Add();
 col.Field("FirstName").Label("First
Name").Type("string").Operators(ViewBag.Operator).Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("MM/dd/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).DataSource(ViewBag.dataSource).Render()
```

### DEFAULT.CS

```
public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 List<object> Operator = new List<object> {
 new { key = "Equal", value = "equal" },
 new { key = "Not Equal", value = "notequal" },
 new { key = "In", value = "in" },
 new { key = "Not In", value = "notin" }
 };
 ViewBag.rule = rule;
 ViewBag.Operator = Operator;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

public class EmployeeView
{
}
```



```

public EmployeeView()
{
}

public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
{
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
}

public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public string Title { get; set; }
public DateTime BirthDate { get; set; }
public DateTime HireDate { get; set; }
public int ReportsTo { get; set; }
public string Address { get; set; }
public string PostalCode { get; set; }
public string Phone { get; set; }
public string City { get; set; }
public string Country { get; set; }
public static List<EmployeeView> GetAllRecords()
{
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "
USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
}

```

```

 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7, "Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "UK"));

 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9, "722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));

 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5, "4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", "London", "UK "));

 return Emp;
 }
}

```

## Validations

Validation allows you to validate the conditions and it display errors for invalid fields while using the `validateFields` method. To enable validation in the query builder, set [AllowValidation](#) to true. Column fields are validated after setting [AllowValidation](#) to true. So, you should manually configure the validation for Operator and Value fields through [Validation](#).

**Note:** Set `isRequired` validation for Operator and Value fields.

Set `min`, `max` values for number values.

## CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}) .DataSource(ViewBag.dataSource).AllowValidation(true).Render()
@Html.EJS().Button("validate").CssClass("e-primary").Content("Validate Fields").Render()

<script>
 document.getElementById('validate').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 querybuilderObj.validateFields();
 }
</script>

```

## DEFAULT.CS

```

public ActionResult Index()
{
 ViewBag.dataSource = EmployeeView.GetAllRecords();
}

```

```

 return View();
 }
}

public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }

 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "
USA"));
 }
}

```

```

 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,
"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK"));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
 }
}

```

## Template

Template allows you to define your own input widgets for columns. To implement the template, you can define the following functions.

- **create**: Creates the custom component.
- **write**: Wire events for the custom component.
- **destroy**: Destroy the custom component.

In the following sample, dropdown is used as the custom component in Payment Mode column.

## CSHTML

```

<div class="col-lg-8 control-section">
 @Html.EJS().QueryBuilder("querybuilder").Columns(col =>
 {
 col.Field("Category").Label("Category").Type("string").Add();
 col.Field("PaymentMode").Label("Payment
Mode").Type("string").Template(ViewBag.template).Operators(ViewBag.paymentOp
erator).Add();
 col.Field("TransactionType").Label("Transaction
Type").Type("boolean").Operators(ViewBag.transactionOperator).Add();

 col.Field("Description").Label("Description").Type("string").Add();
 col.Field("Date").Label("Date").Type("date").Add();

 col.Field("Amount").Label("Amount").Type("number").Operators(ViewBag.amountO
perator).Add();

 }).DataSource(ViewBag.dataSource).Rule(ViewBag.rule).Width("100%").Render()
 </div>
<script>
 function paymentCreate() {
 var elem = document.createElement('input');
 elem.setAttribute('type', 'text');
 return elem;
 }
 function paymentDestroy(args) {

```

```

 var selectObj =
ej.base.getComponent(document.getElementById(args.elementId),
'multiselect');
 if (selectObj) {
 selectObj.destroy();
 }
 var dropdown =
ej.base.getComponent(document.getElementById(args.elementId),
'dropdownlist');
 if (dropdown) {
 dropdown.destroy();
 }
 }
 function paymentWrite(args) {
 var ds = ['Cash', 'Debit Card', 'Credit Card', 'Net Banking',
'Wallet'];
 var inOperators = ['in', 'notin'];
 var qryBldrObj =
ej.base.getComponent(document.getElementById("querybuilder"), 'query-
builder');
 if (inOperators.indexOf(args.operator) > -1) {
 var multiSelectObj = new ej.dropdowns.MultiSelect({
 dataSource: ds,
 value: args.values,
 mode: 'CheckBox',
 placeholder: 'Select Transaction',
 change: function (e) {
 qryBldrObj.notifyChange(e.value, e.element);
 }
 });
 multiSelectObj.appendTo('#' + args.elements.id);
 }
 else {
 var dropDownObj = new ej.dropdowns.DropDownList({
 dataSource: ds,
 value: args.values ? args.values : ds[0],
 change: function (e) {
 qryBldrObj.notifyChange(e.itemData.value, e.element);
 }
 });
 dropDownObj.appendTo('#' + args.elements.id);
 }
 }
</script>
<style>
.e-control-wrapper.e-slider-container.e-horizontal {
 height: 0;
}
.e-querybuilder {
 margin: 3% auto;
}
#ruleContent {
 border: 1px solid #d3d3d3;
 width: 100%;
 height: 500px;
 overflow: auto;
}

```

```
.highcontrast textarea#ruleContent {
 background-color: #000;
}
</style>
```

**DEFAULT.CS**

```
public ActionResult Template()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Category",
Field="Category", Type="string", Operator="equal", Value = new string[] {
"Clothing" } },
 new QueryBuilderRule { Condition="or", Rules = new
List<QueryBuilderRule>() {
 new QueryBuilderRule { Label="Transaction Type",
Field="TransactionType", Type="boolean", Operator="equal", Value = "Income"
},
 new QueryBuilderRule { Label="Payment Mode",
Field="PaymentMode", Type="string", Operator="equal", Value = "Cash" }
 } },
 new QueryBuilderRule { Label="Amount", Field="Amount",
Type="number", Operator="equal", Value = 10 }
 }
 };
 Temp template = new Temp()
 {
 create = "paymentCreate",
 destroy = "paymentDestroy",
 write = "paymentWrite"
 };
 List<object> paymentOperator = new List<object> {
 new { key = "Equal", value = "equal" },
 new { key = "Not Equal", value = "notequal" },
 new { key = "In", value = "in" },
 new { key = "Not In", value = "notin" }
 };
 List<object> transactionOperator = new List<object> {
 new { key = "Equal", value = "equal" },
 new { key = "Not Equal", value = "notequal" }
 };
 List<object> amountOperator = new List<object> {
 new { key = "Equal", value = "equal" },
 new { key = "Greater than", value = "greaterthan" },
 new { key = "Less than", value = "lessthan" },
 new { key = "Less than or equal", value = "lessthanorequal"
},
 new { key = "Greater than or equal", value =
"greaterthanorequal" },
 new { key = "Not equal", value = "notequal" }
 };
 ViewBag.rule = rule;
```

```

 ViewBag.paymentOperator = paymentOperator;
 ViewBag.transactionOperator = transactionOperator;
 ViewBag.amountOperator = amountOperator;
 ViewBag.template = template;
 return View();
 }
 public class Temp
 {
 public string create { get; set; }
 public string destroy { get; set; }
 public string write { get; set; }
 }

```

## Data binding

The Query Builder uses **DataManager** to bind the data source, which supports both RESTful JSON data services binding and local JavaScript object array binding. The [DataSource](#) property can be assigned either with the instance of **DataManager** or JavaScript object array collection. It supports two kinds of data binding method.

- Local data
- Remote data

### Local Data

To bind local data to the query builder, you can assign the [DataSource](#) property with a JavaScript object array. The local data source can also be provided as an instance of the **DataManager**.

### CSHTML

```

@model List<QueryBuilderSample.Controllers.EmployeeView>
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(Model).Render()

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },

```

```

 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}
public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }

 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 }
}

```



```

 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6, "14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8, "Coventry HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2, " 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7, "Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "UK "));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9, "722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5, "4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
 }
}

```

**Note:** By default, DataManager uses JsonAdaptor for local data-binding.

### Remote data

To bind remote data to the query builder, assign service data as an instance of **DataManager** to the [DataSource](#) property. To interact with remote data source, provide the endpoint url.

### CSHTML

```

@Html.EJS().QueryBuilder("RemoteData").DataSource(dataManager =>
{
 dataManger.Url("https://services.syncfusion.com/aspnet/production/api/orders")
 .CrossDomain(true).Adaptor("ODataV4Adaptor");
 }).Columns(col =>
 {
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
 }).Rule(ViewBag.rule).Width("72%").MaxGroupCount(5).Render()

```

### REMOTEDATA.CS

```

public IActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()

```

```

 {
 Condition = "and",
 Rules = new List<QueryBuilderRules>()
 {
 new QueryBuilderRules { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRules { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 return view();
 }
}

```

**Note:** By default, **DataManager** uses **ODataAdaptor** for remote data-binding.

#### *Binding with OData services*

**OData** is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the **DataManager**. Refer to the following code example for remote Data binding using OData service.

#### **CSHTML**

```

@Html.EJS().QueryBuilder("OData").DataSource(dataManger =>
{
 dataManger.Url("https://services.syncfusion.com/aspnet/production/api/orders")
 .CrossDomain(true).Adaptor("ODataAdaptor");
 }).Columns(col =>
 {
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
 }).Rule(ViewBag.rule).Width("72%").MaxGroupCount(5).Render()
}

```

#### **ODATA.CS**

```

public IActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRules>()
 {
 new QueryBuilderRules { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRules { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 }
}

```

```
};
ViewBag.rule = rule;
return view();
}
```

### Binding with ODataV4 services

The ODataV4 is an improved version of OData protocols, and the **DataManager** can also retrieve and consume ODataV4 services. For more details on ODataV4 services, refer to the [odata documentation](#). To bind ODataV4 service, use the **ODataV4Adaptor**.

### CSHTML

```
@Html.EJS().QueryBuilder("OdataV4").DataSource(dataManger =>
{
dataManger.Url("https://services.syncfusion.com/aspnet/production/api/orders")
.CrossDomain(true).Adaptor("ODataV4Adaptor");
}).Columns(col =>
{
col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
col.Field("FirstName").Label("First Name").Type("string").Add();
col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
col.Field("Title").Label("Title").Type("string").Add();
col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
col.Field("Country").Label("Country").Type("string").Add();
col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).Width("72%").MaxGroupCount(5).Render()
```

### ODATAV4.CS

```
public IActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRules>()
 {
 new QueryBuilderRules { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRules { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 return view();
}
```

### Web API

You can use **WebApiAdaptor** to bind query builder with Web API created using OData endpoint.

### CSHTML

```
@Html.EJS().QueryBuilder("WebApi").DataSource(dataManger =>
```

```
{
dataManger.Url("api/OrderAPI").CrossDomain(true).Adaptor("WebApiAdaptor");
}).Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).Width("72%")..MaxGroupCount(5).Render()
```

### WEBAPI.CS

```
public IActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRules>()
 {
 new QueryBuilderRules { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRules { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 return view();
}
```

### Url adaptor

You can use the UrlAdaptor of DataManager when binding data source from remote data. In the initial load of querybuilder, data are fetched from remote data and bound to the querybuilder using url property of DataManager.

### CSHTML

```
@Html.EJS().QueryBuilder("UrlAdaptor").DataSource(dataManger =>
{
dataManger.Url("Home/QBDataSource").CrossDomain(true).Adaptor("UrlAdaptor");
}).Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
```

```
col.Field("Country").Label("Country").Type("string").Add();
col.Field("City").Label("City").Type("string").Add();
}).Width("72%").MaxGroupCount(5).Render()
```

### URL-DATA.CS

```
public IActionResult QBDataSource([FromBody]DataManagerRequest dm)
{
 IEnumerable DataSource = employeeData;
 DataOperations operation = new DataOperations();
 if (dm.Where != null && dm.Where.Count > 0) //Filtering
 {
 DataSource = operation.PerformFiltering(DataSource,
dm.Where, dm.Where[0].Operator);
 }
 int count = DataSource.Cast<OrdersDetails>().Count();
 if (dm.Skip != 0)
 {
 DataSource = operation.PerformSkip(DataSource, dm.Skip);
//Paging
 }
 if (dm.Take != 0)
 {
 DataSource = operation.PerformTake(DataSource, dm.Take);
 }
 return dm.RequiresCounts ? Json(new { result = DataSource, count
= count }) : Json(DataSource);
}
```

### Support with Data Manager

You can use the created conditions in DataManager through the `GetPredicate` method, which results the filtered records.

### CSHTML

```
@Html.EJS().QueryBuilder("querybuilder").Width("850px").Columns(col =>
{
 col.Field("TaskID").Label("Task ID").Type("number").Add();
 col.Field("Name").Label("Name").Type("string").Add();

col.Field("Category").Label("Category").Type("string").Add();
 col.Field("SerialNo").Label("Serial
No").Type("string").Add();
 col.Field("InvoiceNo").Label("Invoice
No").Type("string").Add();
 col.Field("Status").Label("Status").Type("string").Add();
}).Rule(ViewBag.importRules).Render()
@Html.EJS().Button("getpredicate").CssClass("e-primary").Content("Get
Predicate").Render()

<script>
 document.getElementById('getpredicate').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
```

```

 var template =
 '<tr><td>${TaskID}</td><td>${Name}</td><td>${Category}</td></tr>';
 var compileFunction = ej.base.compile(template);
 var table = (document.getElementById('datatable'));
 table.className = 'e-table';
 var predicate =
 querybuilderObj.getPredicate(querybuilderObj.getValidRules(querybuilderObj.rule));
 dataManagerQuery = new ej.data.Query().select(['TaskID', 'Name',
 'Category', 'SerialNo', 'InvoiceNo', 'Status']).where(predicate);
 new
 ej.data.DataManager(@Html.Raw(JsonConvert.SerializeObject(@ViewBag.DataSource)))
 .executeQuery(dataManagerQuery)
 .then(function (e) {
 e.result.forEach(function (data) {
 table.appendChild(compileFunction(data)[0]);
 });
 });
 }
</script>
<style type="text/css">
 .e-hide {
 display: none;
 }
</style>
<body>
 <div id="container">
 <div id="Grid"></div>
 <table id="datatable" class="e-table e-hide">
 <thead>
 <tr><th>Task ID</th><th>Category</th><th>Name</th></tr>
 </thead>
 <tbody>
 </tbody>
 </table>
 </div>
<script>
var ele = document.getElementById('container');
if(ele) {
 ele.style.visibility = "visible";
}
</script>
</body>

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Category",
 Field="Category", Type="string", Operator="equal", Value = "Laptop" }
 }
 }
}

```

```

 }
 };
 ViewBag.rule = rule;
 // hardwareData is referred from MVC sample browser.
 ViewBag.DataSource = QueryBuilderData.hardwareData;
 return View();
}

```

### Grid Integration with QueryBuilder

This section explains how to integrate Grid with QueryBuilder. We have used **UrlAdaptor** to load dataSource to Grid, and the dataSource property is optional for QueryBuilder. So, you can create QueryBuilder with only columns. QueryBuilder interacts with Grid through the **Query** property of a Grid. You can get the query from QueryBuilder and update the Grid Query property in the **ruleChange** event of QueryBuilder. The **UrlAdaptor** works based on the on-demand concept, it will load only current page records from the server and request the next page records when navigating to the next page. Also, we have sent the request to the server for every action performed in the Grid.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("73%").Columns(col =>
{
 col.Field("CustomerID").Label("Customer ID").Type("number").Add();
 col.Field("CompanyName").Label("Company Name").Type("string").Add();
 col.Field("ContactName").Label("Contact Name").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).RuleChange("updateRule").Render()
@Html.EJS().Grid("Grid").DataSource(dataManager => {
dataManager.Url("/Home/UrlDataSource").Adaptor("UrlAdaptor"); }).Columns(col
=>
{
 col.Field("CustomerID").HeaderText("Customer ID").Width("120").Add();
 col.Field("CompanyName").HeaderText("Company Name").Width("120").Add();
 col.Field("ContactName").HeaderText("Contact Name").Width("120").Add();
 col.Field("City").HeaderText("City").Width("120").Add();
}).AllowPaging().Render()
<script>
 function updateRule(args) {
 var dataManagerQuery;
 var qryBldrObj =
ej.base.getComponent(document.getElementById("querybuilder"), 'query-
builder');
 var grid = ej.base.getComponent(document.getElementById("grid"),
'grid');
 var predicate = qryBldrObj.getPredicate(args.rule);
 if (ej.base.isNullOrUndefined(predicate)) {
 dataManagerQuery = new ej.data.Query().select(['CustomerID',
'CompanyName', 'ContactName', 'City']);
 } else {
 dataManagerQuery = new ej.data.Query().select(['CustomerID',
'CompanyName', 'ContactName', 'City']).where(predicate);
 }
 grid.query = dataManagerQuery;
 grid.refresh();
 }
</script>

```

**HOMECONTROLLER.CS**

```

public class HomeController : Controller
{
 private NORTHWNDContext _context;
 public HomeController(NORTHWNDContext context)
 {
 _context = context;
 }
 public IActionResult Index()
 {
 return View();
 }
 private IQueryable<Records> GetDataSource()
 {
 IQueryable<Records> transactions = _context.Records;
 return transactions;
 }
 public IActionResult UrlDatasource([FromBody] DataManagerRequest dm)
 {
 IQueryable<Records> DataSource = GetDataSource();
 QueryableOperation operation = new QueryableOperation();
 if (dm.Where != null && dm.Where.Count > 0) //Filtering
 {
 DataSource = operation.PerformFiltering(DataSource, dm.Where,
dm.Where[0].Condition); //filtering
 }
 if (dm.Search != null && dm.Search.Count > 0)
 {
 DataSource = operation.PerformSearching(DataSource, dm.Search);
//Search
 }
 int count = DataSource.Cast<Records>().Count();
 if (dm.Skip != 0)
 {
 DataSource = operation.PerformSkip(DataSource, dm.Skip);
 }
 if (dm.Take != 0)
 {
 DataSource = operation.PerformTake(DataSource, dm.Take);
 }
 return dm.RequiresCounts ? Json(new { result = DataSource, count =
count }) : Json(DataSource);
 }
}

```

**Model Binding Support**

Model binding allows to bind properties for the components used in field, operator, and value columns. To implement model binding, assign [FieldModel](#), [OperatorModel](#), and [ValueModel](#) properties in QueryBuilder.

**CSHTML**

```
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
```



```

{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
})
).FieldModel(ViewBag.fieldModel).OperatorModel(ViewBag.operatorModel).Value
Model(ViewBag.valueModel).Render()

```

### DEFAULT.CS

```

public ActionResult Index()
{
 List<string> values = new List<string> { "Mr.", "Mrs." };
 ViewBag.values = values;
 var fieldModel = new { allowFiltering = true, popupHeight =
"380px" };
 var operatorModel = new { allowFiltering = true, popupHeight =
"380px" };
 var valueModel = new { numericTextBoxModel = new { cssClass =
"e-custom" } };
 ViewBag.fieldModel = fieldModel;
 ViewBag.operatorModel = operatorModel;
 ViewBag.valueModel = valueModel;
 return View();
}

```

### Filtering

Query Builder allows you to create or delete conditions and groups. You can use **ShowButtons** to enable/disable these buttons.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).ShowButtons(new
Syncfusion.EJ2.QueryBuilder.QueryBuilderShowButtons { GroupDelete = true,
GroupInsert = true, RuleDelete = true }).Render()
@Html.EJS().Button("addgroup").CssClass("e-primary").Content("Add
Group").Render()

```

```

@Html.EJS().Button("addrule").CssClass("e-primary").Content("Add
Rule").Render()
@Html.EJS().Button("deletegroup").CssClass("e-primary").Content("Delete
Group").Render()

<script>
 document.getElementById('addgroup').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 querybuilderObj.addGroups([{ 'condition': 'and', 'rules': [{
'label': 'First Name', 'field': 'FirstName', 'type': 'string', 'operator':
'startswith', 'value': 'v' }] }], 'group0');
 }
 document.getElementById('addrule').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 querybuilderObj.addRules([{ 'label': 'City', 'field': 'City',
'type': 'string', 'operator': 'equal', 'value': 'US' }], 'group0');
 }
 document.getElementById('deletegroup').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 querybuilderObj.deleteGroups(['group0']);
 }
</script>

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int

```

```

ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
{
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
}

public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public string Title { get; set; }
public DateTime BirthDate { get; set; }
public DateTime HireDate { get; set; }
public int ReportsTo { get; set; }
public string Address { get; set; }
public string PostalCode { get; set; }
public string Phone { get; set; }
public string City { get; set; }
public string Country { get; set; }
public static List<EmployeeView> GetAllRecords()
{
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "
USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,
"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK"));
}

```

```

 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", " (206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
 }
}

```

You can create or delete conditions by interacting through the user interface and methods.

- Use the `addRules` and `deleteRules` methods to create/delete conditions.
- Use the `addGroups` and `deleteGroups` methods to create/delete groups.

## Importing

Importing allows you to view or edit the predefined conditions. You can import the conditions by using methods and initial rendering.

### Initial Rendering

To apply conditions initially, you can define the `Rule`. Here, you can import structured JSON object by defining the `Rule` property.

### CSHTML

```

@model List<QueryBuilderSample.Controllers.EmployeeView>
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(Model).Render()

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
}

```

```

 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
 }
}

public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }

 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,

```

```

"Coventry HouseMiner Rd. ", "EC2 7JR ", " (206) 555-8122", "Tacoma ", "
USA")));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,
"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK"));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", " (206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
}
}

```

## Post Rendering

### Import from JSON

You can set the conditions from structured JSON object through the `setRules` method.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Render()
@Html.EJS().Button("importrules").CssClass("e-primary").Content("Import
Rule").Render()

<script>
 document.getElementById('importrules').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 querybuilderObj.setRules({ 'condition': 'or', 'rules': [{ 'label':
'Employee ID', 'field': 'EmployeeID', 'type': 'number', 'operator': 'equal',
'value': '1001' }] });
 }
</script>

```

### DEFAULT.CS

```

public ActionResult Index()
{

```

```

 return View();
}

```

### Import From SQL

You can set the conditions from SQL query through the `setRulesFromSql` method.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("EmployeeID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}) .Render()
@Html.EJS().Button("importrules").CssClass("e-primary").Content("Import
Rule").Render()
<script>
 document.getElementById('importrules').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 querybuilderObj.setRulesFromSql("EmployeeID = 1");
 }
</script>

```

### DEFAULT.CS

```

public ActionResult Index()
{
 return View();
}

```

### Exporting

Exporting allows you to save or maintain the created conditions through the Query Builder. You can export the defined conditions by the following ways.

#### JSON Export

You can export the defined conditions to structured JSON object through the `getRules` method.

#### SQL Support

You can export the defined conditions to SQL query through the `getRulesFromSQL` method.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
}

```

```

 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
 }).Render()

@Html.EJS().Dialog("defaultdialog").Target(".e-query-
builder").Width("50%").Header("Query
builder").Visible(false).ShowCloseIcon(true).Render()
@Html.EJS().Button("getsql").CssClass("e-primary").Content("Get
Sql").Render()
@Html.EJS().Button("getrule").CssClass("e-primary").Content("Get
Rule").Render()
<script>
 document.getElementById('getsql').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 var dialogObj =
ej.base.getInstance(document.getElementById("defaultdialog"),
ej.popups.Dialog);
 dialogObj.content =
querybuilderObj.getSqlFromRules(querybuilderObj.getRules());
 dialogObj.show();
 }
 document.getElementById('getrule').onclick = function () {
 var querybuilderObj =
ej.base.getInstance(document.getElementById("querybuilder"),
ej.querybuilder.QueryBuilder);
 var dialogObj =
ej.base.getInstance(document.getElementById("defaultdialog"),
ej.popups.Dialog);
 dialogObj.content = '<pre>' +
JSON.stringify(querybuilderObj.getValidRules(querybuilderObj.rule), null, 4)
+ '</pre>';
 dialogObj.show();
 }
</script>

```

## DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 }
}

```



```
};
ViewBag.rule = rule;
return View();
}
```

## Localization

The **Localization** library allows you to localize the default text content of the Query Builder. The Query Builder has static text that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the **Locale** value and translation object.

The following list of properties and its values are used in the Query Builder.

|                    |                        |
|--------------------|------------------------|
| Locale key words   | Text                   |
| -----              | -----                  |
| AddGroup           | Add Group              |
| AddCondition       | Add Condition          |
| DeleteRule         | Remove this condition  |
| DeleteGroup        | Delete group           |
| Edit               | EDIT                   |
| SelectField        | Select a field         |
| SelectOperator     | Select operator        |
| StartsWith         | Starts With            |
| EndsWith           | Ends With              |
| Contains           | Contains               |
| Equal              | Equal                  |
| NotEqual           | Not Equal              |
| LessThan           | Less Than              |
| LessThanOrEqual    | Less Than Or Equal     |
| GreaterThan        | Greater Than           |
| GreaterThanOrEqual | Greater Than Or Equal  |
| Between            | Between                |
| NotBetween         | Not Between            |
| In                 | In                     |
| NotIn              | Not In                 |
| Remove             | REMOVE                 |
| ValidationMessage  | This field is required |
| SummaryViewTitle   | Summary View           |

| OtherFields | Other Fields |

| AND | AND |

| OR | OR |

| SelectValue | Enter Value |

| IsEmpty | Is Empty |

| IsNotEmpty | Is Not Empty |

| IsNull | Is Null |

| IsNotNull | Is Not Null |

**CSHTML**

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).DataSource(ViewBag.dataSource).Locale("de-
DE").Render()
<script type="text/javascript">
var qbObj = document.getElementById('querybuilder1').ej2_instances[0];
ej.base.L10n.load({
 'de-DE': {
 'querybuilder': {
 'AddGroup': 'Gruppe hinzufügen',
 'AddCondition': 'Bedingung hinzufügen',
 'DeleteRule': 'Entfernen Sie diesen Zustand',
 'DeleteGroup': 'Gruppe löschen',
 'Edit': 'BEARBEITEN',
 'SelectField': 'Wählen Sie ein Feld aus',
 'DeleteRule': 'Entfernen Sie diesen Zustand',
 'DeleteGroup': 'Gruppe löschen',
 'SelectOperator': 'Operator auswählen',
 'StartsWith': 'Beginnt mit',
 'EndsWith': 'Endet mit',
 'Contains': 'Enthält',
 'Equal': 'Gleich',
 'NotEqual': 'Nicht gleich',
 'LessThan': 'Weniger als',
 'LessThanOrEqual': 'Weniger als oder gleich',
 'GreaterThan': 'Größer als',
 'GreaterThanOrEqual': 'Größer als oder gleich',
 'Between': 'Zwischen',
 'NotBetween': 'Nicht zwischen',
 'In': 'Im',
 'NotIn': 'Nicht in',
 'Remove': 'LÖSCHEN',

```

```

 'ValidationMessage': 'Dieses Feld wird benötigt',
 }
}
});
</script>

```

## DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };

 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

public class EmployeeView
{
 public EmployeeView()
 {
 }

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }

 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
}

```

```

public string Address { get; set; }
public string PostalCode { get; set; }
public string Phone { get; set; }
public string City { get; set; }
public string Country { get; set; }
public static List<EmployeeView> GetAllRecords()
{
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "
USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,
"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK"));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
}
}

```

## Styles and Appearances

To modify the QueryBuilder appearance, you need to override the default CSS of QueryBuilder component. Find the list of CSS classes and its corresponding section in QueryBuilder component. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

### CSS Class | Purpose of Class

|.e-group-header .e-btn|To customize the condition button in querybuilder

|.e-group-body .e-rule-container|To customize the querybuilder rule container

|.e-group-container .e-group-header .e-dropdown-btn|To customize the querybuilder Add group/condition button

`|.e-query-builder .e-group-header .e-deletegroup|` To customize the querybuilder Delete group button

`|.e-query-builder .e-rule-field .e-rule-value-delete .e-rule-delete|` To customize the querybuilder Delete condition button

`|.e-query-builder .e-rule-list > ::after, .e-query-builder .e-rule-list > ::before|` To customize the querybuilder group joining line

`|.e-query-builder .e-rule-container.e-joined-rule|` To customize the querybuilder condition joining line

## Templates

### Header Template

Header Template allows to define your own user interface for Header, which includes creating or deleting rules and groups and to customize the AND/OR condition and NOT condition options. To implement header template in querybuilder, you can create the user interface using `x-template` and assign the values when `requestType` is `header-template-create` in `actionBegin` event.

### CSHTML

```
@Html.EJS().QueryBuilder("querybuilder").Width("850px").HeaderTemplate("#headerTemplate").ActionBegin("actionBegin").EnableNotCondition(true).Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("LastName").Label("LastName").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.importRules).MaxGroupCount(5).Render()
</div>
</div>
</div>
<style>
.e-query-builder .e-add-btn {
 margin-left: 10px;
 margin-right: 10px;
}
.e-query-builder .cndtnbtn.e-control.e-dropdownlist.e-lib.e-input {
 padding-left: 10px;
}
.e-query-builder span.e-custom-group-btn {
 max-width: 100px;
 border-radius: 5px !important;
 border-width: 1px !important;
 margin-top: 3px;
}
.e-query-builder .e-custom-group-btn.e-input-focus::before, .e-custom-group-btn.e-input-focus::after {
 background: transparent !important;
}
```

```

 }
 .e-query-builder .e-group-header .e-addrulegroup, .e-group-header .e-
delete-btn {
 border: 1px solid grey !important;
 }
 .e-query-builder .e-group-header .e-addrulegroup:hover, .e-group-header
.e-delete-btn:hover {
 border: 1px solid grey !important;
 }
 .e-query-builder .e-toggle {
 background: #317ab9;
 border-color: #317ab9;
 color: #fff;
 }
 .e-query-builder .e-cb-wrapper {
 margin-right: 5px;
 height: 32px;
 border-radius: 5px;
 border: 1px solid gray;
 background-color: white;
 }
 .e-query-builder .e-custom-group-btn {
 padding-left: 10px;
 height: 32px;
 }
}
</style>
<script id="headerTemplate" type="text/x-template">
 <div class='e-groupheader'>
 ${if(notCondition !== undefined)}
 <button class="e-cb-wrapper">
 <input type="checkbox" class="e-not" id='${ruleID}_notoption'
checked='${notCondition}'>
 </button>
 ${/if}
 <input type="text" class="e-custom-group-btn"
id='${ruleID}_cndtnbtn'>
 <button id='${ruleID}_addbtn' class='e-add-btn'></button>
 ${if(ruleID !== 'querybuilder_group0')}
 <button id='dltbtn' class="e-btn e-delete-btn e-lib e-small e-round
e-icon-btn">

 </button>
 ${/if}
 </div>
</script>
<script>
function actionBegin(args) {
 debugger
 var queryBldrObj =
ej.base.getComponent(document.getElementById("querybuilder"), "query-
builder");
 if (args.requestType === 'header-template-create') {
 var checkBoxObj = new ej.buttons.CheckBox({
 label: 'NOT',
 checked: args.notCondition,
 change: function (e) {

```

```

 queryBldrObj.notifyChange(e.checked, e.event.target,
'not');
 }
 });
 checkBoxObj.appendTo('#' + args.ruleID + '_notoption');
 var ds = [{ 'key': 'AND', 'value': 'and' }, { 'key': 'OR',
'value': 'or' }];
 var btnObj = new ej.dropdowns.DropDownList({
 dataSource: ds,
 fields: { text: 'key', value: 'value' },
 value: args.condition,
 cssClass: 'e-custom-group-btn e-active-toggle',
 change: function (e) {
 queryBldrObj.notifyChange(e.value, e.element,
'condition');
 }
 });
 btnObj.appendTo('#' + args.ruleID + '_cndtnbtn');
 var ddbitems = [
 { text: 'AddGroup', iconCss: 'e-icons e-add-icon e-addgroup'
},
 { text: 'AddCondition', iconCss: 'e-icons e-add-icon e-
addrule' }
];
 var addbtn = new ej.splitbuttons.DropDownButton({
 items: ddbitems,
 cssClass: 'e-round e-small e-caret-hide e-addrulegroup',
 iconCss: 'e-icons e-add-icon',
 select: function (event) {
 var addbtn = ej.base.closest(event.element, '.e-
dropdown-popup');
 var ddb = addbtn.id.split('_');
 if (event.item.text === 'AddGroup') {
 queryBldrObj.addGroups([{ condition: 'and', 'rules':
[{}], not: false }], ddb[1]);
 }
 else if (event.item.text === 'AddCondition') {
 queryBldrObj.addRules([{}], ddb[1]);
 }
 }
 });
 addbtn.appendTo('#' + args.ruleID + '_addbtn');
 var deleteGroupBtn =
document.getElementById(args.ruleID).querySelector('.e-delete-btn');
 if (deleteGroupBtn) {
 deleteGroupBtn.onclick = function (e) {
 queryBldrObj.deleteGroup(ej.base.closest(e.target.offsetParent, '.e-group-
container'));
 }
 }
}
}
</script>

```

**DEFAULT.CS**

```

public IActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Not = true,
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
 Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="LastName", Field="LastName",
 Type="string", Operator="equal", Value = "MALAN" },
 {
 new QueryBuilderRule {
 Condition = "or",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="LastName",
 Field="LastName", Type="string", Operator="equal", Value = "MALAN" }
 }
 }
 }
 }
 };
 ViewBag.rule = rule;
 return View();
}

```

### Accessibility in Query builder control

The Query Builder component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Query Builder component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |



```

| Keyboard Navigation Support | |
| Accessibility Checker Validation | |
| Axe-core Accessibility Validation | |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

WAI-ARIA (Accessibility Initiative – Accessible Rich Internet Applications) defines a way to increase the accessibility of web pages, dynamic content, and user interface components developed with Ajax, HTML, JavaScript, and related technologies. ARIA provides additional semantics to describe the role, state, and functionality of web components.

The following list of ARIA attributes is used in Query Builder.

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the query builder component. |

### Keyboard interaction

The Query Builder component followed the keyboard interaction guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Query Builder component.

| **Press** | **To do this** |

| --- | --- |

| **Tab / Shift + Tab** | **To focus the next item in the rule.** |

### Ensuring accessibility

The Query Builder component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Query Builder component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Query Builder component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

## How To

### Display Options

Display options allow you to view the Query Builder Vertically or Horizontally. For this, you should use the `DisplayMode` property.

### CSHTML

```
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(ViewBag.dataSource).Rule(ViewBag.rule).DisplayMode(Syncfusion.
EJ2.QueryBuilder.DisplayMode.Vertical).Render()
```

### DEFAULT.CS

```
public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

public class EmployeeView
{
 public EmployeeView()
 {
 }
}
```

```

 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }
 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "
USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,
"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK"));
 }

```

```

 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", " (206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
 }
}

```

**Note:** The default view in the desktop mode is Horizontal.

<br/> The default view in the mobile mode is Vertical.

### Sort the columns

SortDirection allows you to sort the columns bounded to the Query Builder to view the columns by ascending or descending order. You should set the **SortDirection** property to sort the fields.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
})
.DataSource(ViewBag.dataSource).Rule(ViewBag.rule).SortDirection(Syncfusio
n.EJ2.QueryBuilder.SortDirection.Ascending).Render()

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

public class EmployeeView
{

```

```

public EmployeeView()
{
}

public EmployeeView(int EmployeeID, string FirstName, string LastName,
string Title, DateTime BirthDate, DateTime HireDate, int ReportsTo, string
Address, string PostalCode, string Phone, string City, string Country)
{
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
}

public int EmployeeID { get; set; }
public string FirstName { get; set; }
public string LastName { get; set; }
public string Title { get; set; }
public DateTime BirthDate { get; set; }
public DateTime HireDate { get; set; }
public int ReportsTo { get; set; }
public string Address { get; set; }
public string PostalCode { get; set; }
public string Phone { get; set; }
public string City { get; set; }
public string Country { get; set; }
public static List<EmployeeView> GetAllRecords()
{
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales Manager",
new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8, "Coventry
HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", " USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,

```

```

"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK")));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
}
}

```

### Restrict the groups

You can restrict the groups by defining the `MaxGroupCount` property. By default, the value is 5. In the below demo, the `MaxGroupCount` is set to 2 .

### CSHTML

```

@model List<QueryBuilderSample.Controllers.EmployeeView>
@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(Model).Render()

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID",
Field="EmployeeID", Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

public class EmployeeView
{
 public EmployeeView()
 {

```

```

 }
 public EmployeeView(int EmployeeID, string FirstName, string
LastName, string Title, DateTime BirthDate, DateTime HireDate, int
ReportsTo, string Address, string PostalCode, string Phone, string City,
string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }
 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales
Manager", new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8,
"Coventry House Miner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", "
USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales
Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7,

```

```

"Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "
UK")));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales
Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9,
"722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales
Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5,
"4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", " London", "UK "));
 return Emp;
}
}

```

**Note:** You can use this property in the mobile mode to restrict the nested group creation.

### State Persistence

State persistence allows you to maintain the current state in the browser's `localStorage` even if the browser is refreshed or if you move to the next page within the browser. State persistence stores the Query Builder's `Rule` object in the local storage when the `EnablePersistence` is defined to true.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).DataSource(ViewBag.dataSource).Rule(ViewBag.rule).EnablePersistence(true)
.Render()

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 ViewBag.dataSource = EmployeeView.GetAllRecords();
 return View();
}

```



```

public class EmployeeView
{
 public EmployeeView()
 {
 }
 public EmployeeView(int EmployeeID, string FirstName, string LastName,
string Title, DateTime BirthDate, DateTime HireDate, int ReportsTo, string
Address, string PostalCode, string Phone, string City, string Country)
 {
 this.EmployeeID = EmployeeID;
 this.FirstName = FirstName;
 this.LastName = LastName;
 this.Title = Title;
 this.BirthDate = BirthDate;
 this.HireDate = HireDate;
 this.ReportsTo = ReportsTo;
 this.Address = Address;
 this.PostalCode = PostalCode;
 this.Phone = Phone;
 this.City = City;
 this.Country = Country;
 }
 public int EmployeeID { get; set; }
 public string FirstName { get; set; }
 public string LastName { get; set; }
 public string Title { get; set; }
 public DateTime BirthDate { get; set; }
 public DateTime HireDate { get; set; }
 public int ReportsTo { get; set; }
 public string Address { get; set; }
 public string PostalCode { get; set; }
 public string Phone { get; set; }
 public string City { get; set; }
 public string Country { get; set; }
 public static List<EmployeeView> GetAllRecords()
 {
 List<EmployeeView> Emp = new List<EmployeeView>();
 Emp.Add(new EmployeeView(1, "Nancy", "Davolio", "Sales
Representative", new DateTime(1948, 12, 08), new DateTime(1992, 05, 01), 2,
"507 - 20th Ave. E.Apt. 2A ", " 98122", "(206) 555-9857 ", "Seattle ",
"USA"));
 Emp.Add(new EmployeeView(2, "Andrew", "Fuller", "Vice President,
Sales", new DateTime(1952, 02, 19), new DateTime(1992, 08, 14), 4, "908 W.
Capital Way", "98401 ", "(206) 555-9482 ", "Kirkland ", "USA"));
 Emp.Add(new EmployeeView(3, "Janet", "Leverling", "Sales
Representative", new DateTime(1963, 08, 30), new DateTime(1992, 04, 01), 3,
" 4110 Old Redmond Rd.", "98052 ", "(206) 555-8122", "Redmond ", "USA "));
 Emp.Add(new EmployeeView(4, "Margaret", "Peacock", "Sales
Representative", new DateTime(1937, 09, 19), new DateTime(1993, 05, 03), 6,
"14 Garrett Hill ", "SW1 8JR ", "(71) 555-4848 ", "London ", "UK "));
 Emp.Add(new EmployeeView(5, "Steven", "Buchanan", "Sales Manager",
new DateTime(1955, 03, 04), new DateTime(1993, 10, 17), 8, "Coventry
HouseMiner Rd. ", "EC2 7JR ", "(206) 555-8122", "Tacoma ", " USA"));
 Emp.Add(new EmployeeView(6, "Michael", "Suyama", "Sales
Representative", new DateTime(1963, 07, 02), new DateTime(1993, 10, 17), 2,
" 7 Houndstooth Rd.", " WG2 7LT", "(71) 555-4444 ", "London ", "UK "));
 }
}

```

```

 Emp.Add(new EmployeeView(7, "Robert", "King", "Sales Representative", new DateTime(1960, 05, 29), new DateTime(1994, 01, 02), 7, "Edgeham HollowWinchester Way ", "RG1 9SP ", "(71) 555-5598 ", "London ", "UK"));
 Emp.Add(new EmployeeView(8, "Laura", "Callahan", "Inside Sales Coordinator", new DateTime(1958, 01, 09), new DateTime(1994, 03, 05), 9, "722 Moss Bay Blvd. ", "98033 ", "(206) 555-3412", "Seattle ", "USA "));
 Emp.Add(new EmployeeView(9, "Anne", "Dodsworth", "Sales Representative", new DateTime(1966, 01, 27), new DateTime(1994, 11, 15), 5, "4726 - 11th Ave. N.E. ", "98105 ", "(71) 555-5598 ", "London", "UK"));
 return Emp;
}
}

```

### Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Query Builder component from right-to-left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL, set the `EnableRtl` to true.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).EnableRtl(true).Locale("ar-AE").Render()
<script type="text/javascript">
var qbObj = document.getElementById('querybuilder1').ej2_instances[0];
ej.base.L10n.load({
 'ar-AE': {
 'querybuilder': {
 'AddGroup': 'إضافة مجموعة',
 'AddCondition': 'إضافة الشرط',
 'DeleteRule': 'أزل هذا الشرط',
 'DeleteGroup': 'حذف المجموعة',
 'Edit': 'تصحيح',
 'SelectField': 'اختر مجال',
 'DeleteRule': 'أزل هذا الشرط',
 'DeleteGroup': 'حذف المجموعة',
 'SelectOperator': 'حدد المشغل',
 'StartsWith': 'أبدأ بـ',
 'EndsWith': 'ينتهي مع',
 'Contains': 'يحتوي على',
 'Equal': 'متساو',
 'NotEqual': 'ليس متساوي',
 'LessThan': 'أقل من',
 'LessThanOrEqual': 'أصغر من أو يساوي',
 'GreaterThan': 'أكبر من',
 'GreaterThanOrEqual': 'أكبر من أو يساوي',

```

```

 'Between': 'ما بين',
 'NotBetween': 'ليس بينهما',
 'In': 'في',
 'NotIn': 'ليس في',
 'Remove': 'إزالة',
 'ValidationMessage': 'هذه الخانة مطلوبة'
 }
}
});
</script>

```

### DEFAULT.CS

```

public ActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {
 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 return View();
}

```

### Summary View

Summary view allows you to show or hide the filtered query. By default, the value is false. You can enable by setting the `summaryView` property to true.

### CSHTML

```

@Html.EJS().QueryBuilder("querybuilder").Width("72%").Columns(col =>
{
 col.Field("EmployeeID").Label("Employee ID").Type("number").Add();
 col.Field("FirstName").Label("First Name").Type("string").Add();
 col.Field("TitleOfCourtesy").Label("Title Of
Courtesy").Type("boolean").Values(new List<string> { "Mr.", "Mrs." }).Add();
 col.Field("Title").Label("Title").Type("string").Add();
 col.Field("HireDate").Label("Hire
Date").Type("date").Format("dd/MM/yyyy").Add();
 col.Field("Country").Label("Country").Type("string").Add();
 col.Field("City").Label("City").Type("string").Add();
}).Rule(ViewBag.rule).SummaryView(true).Render()

```

### DEFAULT.CS

```

public IActionResult Index()
{
 QueryBuilderRule rule = new QueryBuilderRule()
 {

```

```

 Condition = "and",
 Rules = new List<QueryBuilderRule>()
 {
 new QueryBuilderRule { Label="Employee ID", Field="EmployeeID",
Type="number", Operator="equal", Value = 1 },
 new QueryBuilderRule { Label="Title", Field="Title",
Type="string", Operator="equal", Value = "Sales Manager" }
 }
 };
 ViewBag.rule = rule;
 return View();
}

```

## RadioButton

### Getting Started with ASP.NET MVC Radio Button Control

This section briefly explains about how to include [ASP.NET MVC Radio Button](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

<namespaces>

<add namespace="Syncfusion.EJ2"/>

```
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Radio Button control

Now, add the Syncfusion ASP.NET MVC Radio Button control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Option 1").Name("default").Render()
@Html.EJS().RadioButton("radio2").Label("Option 2").Name("default").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Radio Button control will be rendered in the default web browser.

☐ Option 1 ☒ Option 2 .

### Change the Radio Button state

The Radio Button contains 2 states visually, they are as follows:

- Checked
- Unchecked

The Radio Button [Checked](#) property is used to handle the checked and unchecked state. In the checked state an inner circle will be added to the visualization of Radio Button.

#### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Option
1").Name("state").Checked(true).Render()
@Html.EJS().RadioButton("radio2").Label("Option 2").Name("state").Render()
```

**Note:** [View Sample in GitHub.](#)

**Note:** You can also explore our [ASP.NET MVC Radio Button Example](#) that shows you how to render the Radio Button in ASP.NET MVC.

### Label and Size in Radio Button Component

This section explains the different sizes and labels.

#### Label

RadioButton caption can be defined by using the [label](#) property.

This reduces the manual addition of label for RadioButton. You can customize the label position before or after the RadioButton through the [labelPosition](#) property.

#### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Option
1").Name("state").Checked(true).Render()
@Html.EJS().RadioButton("radio2").Label("Option 2").Name("state").Render()
<style>
.e-radio-wrapper {
 margin-top: 18px;
}
li {
 list-style: none;
}
</style>
```

#### LABEL.CS

```
public ActionResult Label()
{
 return View();
}
```

Left Side Label ☒

☐ Right Side Label

### Size

The different RadioButton sizes available are default and small. To reduce the size of the default RadioButton to small, set the [cssClass](#) property to `e-small`.

### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Small").Name("size").CssClass("e-small").Checked(true).Render()
@Html.EJS().RadioButton("radio2").Label("Default").Name("size").Render()
<style>
.e-radio-wrapper {
 margin-top: 18px;
}
li {
 list-style: none;
}
</style>
```

### SIZE.CS

```
public ActionResult Size()
{
 return View();
}
```

☒ Small

☐ Default

### See Also

- [How to customize the RadioButton appearance](#)

### Styles and Appearances

To modify the RadioButton appearance, you need to override the default CSS of RadioButton component. Find the list of CSS classes and its corresponding section in RadioButton. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

#### CSS Class | Purpose of Class


`|.e-radio-wrapper|` To customize the radio button wrapper

`|.e-radio + label:hover::before|` To customize the radiobutton on hover


`|.e-radio:checked + label::after, e-radio:checked + label::before |` To customize the checked radiobutton

`|.e-radio:checked:focus + label::before, .e-radio:checked + label:hover::before |` To customize the checked radiobutton on hover

 Primary

 Success

 Info

 Warning

 Danger

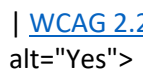
### Accessibility in Radio Button Component

The Radio button component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Radio button component is outlined below.

| Accessibility Criteria | Compatibility |

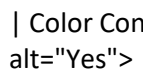
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Yes"> |

| Mobile Device Support |  alt="Yes"> |

| Keyboard Navigation Support |  alt="Yes"> |

| [Accessibility Checker](#) Validation |  alt="Yes"> |

| [Axe-core](#) Accessibility Validation |  alt="Yes"> |

<style>

.post .post-content img {



```
display: inline-block;
margin: 0.5em 0;
}
```

```
</style>
```

```
<div> - All
features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

### WAI-ARIA attributes

The Radio button component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Radio button component:

| Attributes | Purpose |
|------------|---------|
|------------|---------|

|     |     |
|-----|-----|
| --- | --- |
|-----|-----|

|               |                                                                                                      |
|---------------|------------------------------------------------------------------------------------------------------|
| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |
|---------------|------------------------------------------------------------------------------------------------------|

### Keyboard interaction

The Radio button component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Radio button component.

| Press | To do this |
|-------|------------|
|-------|------------|

|     |     |
|-----|-----|
| --- | --- |
|-----|-----|

|               |                                       |
|---------------|---------------------------------------|
| UP/Left arrow | Move and select the previous options. |
|---------------|---------------------------------------|

|                  |                                   |
|------------------|-----------------------------------|
| Down/Right arrow | Move and select the next options. |
|------------------|-----------------------------------|

### Ensuring accessibility

The Radio button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Radio button component is shown in the following sample. Open the [sample](https://ej2.syncfusion.com/accessibility/Radio button.html) in a new window to evaluate the accessibility of the Radio button component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

### How To

#### Customize RadioButton Appearance

You can customize the appearance of the RadioButton component by using the CSS rules.

Define own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

The background and border color of the RadioButton is customized through the custom classes to create the primary, success, info, warning, and danger type of RadioButton.

### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Primary").Name("custom").CssClass("e-primary").Render()
@Html.EJS().RadioButton("radio2").Label("Success").Name("custom").CssClass("e-success").Render()
@Html.EJS().RadioButton("radio3").Label("Info").Name("custom").CssClass("e-info").Checked(true).Render()
@Html.EJS().RadioButton("radio4").Label("Warning").Name("custom").CssClass("e-warning").Render()
@Html.EJS().RadioButton("radio5").Label("Danger").Name("custom").CssClass("e-danger").Render()
<style>
.e-radio-wrapper {
 margin-top: 18px;
}
li {
 list-style: none;
}
.e-success .e-radio:checked + label::after { /* csslint allow: adjoining-classes */
 background-color: #689f38;
 border-color: #689f38;
}
.e-success .e-radio:checked:focus + label::after, .e-success .e-radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
 background-color: #449d44;
 border-color: #449d44;
}
.e-success .e-radio:checked + label::before {
 border-color: #689f38;
}
.e-success .e-radio:checked:focus + label::before, .e-success .e-radio:checked + label:hover::before { /* csslint allow: adjoining-classes */
 border-color: #449d44;
}
.e-success .e-radio + label:hover::before {
 border-color: #blafaf
}
.e-info .e-radio:checked + label::after { /* csslint allow: adjoining-classes */
 background-color: #2196f3;
 border-color: #2196f3;
}
.e-info .e-radio:checked:focus + label::after, .e-info .e-radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
 background-color: #0b7dda;
 border-color: #0b7dda;
}
.e-info .e-radio:checked + label::before {
 border-color: #2196f3;
}
```

```

.e-info .e-radio:checked:focus + label::before, .e-info .e-radio:checked +
label:hover::before {
 border-color: #0b7dda;
}
.e-info .e-radio + label:hover::before {
 border-color: #b1afaf
}
.e-warning .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
 background-color: #ef6c00;
 border-color: #ef6c00;
}
.e-warning .e-radio:checked:focus + label::after, .e-warning .e-
radio:checked + label:hover::after { /* csslint allow: adjoining-classes */
 background-color: #cc5c00;
}
.e-radio:checked + .e-warning::before {
 border-color: #ef6c00;
}
.e-warning .e-radio:checked:focus + label::before, .e-warning .e-
radio:checked + label:hover::before {
 border-color: #cc5c00;
}
.e-warning .e-radio + label:hover::before {
 border-color: #b1afaf
}
.e-danger .e-radio:checked + label::after { /* csslint allow: adjoining-
classes */
 background-color: #d84315;
 border-color: #d84315;
}
.e-danger .e-radio:checked:focus + label::after, .e-danger .e-radio:checked
+ label:hover::after { /* csslint allow: adjoining-classes */
 background-color: #ba330a;
 border-color: #ba330a;
}
.e-danger .e-radio:checked + label::before {
 border-color: #d84315;
}
.e-danger .e-radio:checked:focus + label::before, .e-danger .e-radio:checked
+ label:hover::before {
 border-color: #ba330a;
}
.e-danger .e-radio + label:hover::before {
 border-color: #b1afaf
}
</style>

```

### CUSTOM.CS

```

public ActionResult Custom()
{
 return View();
}

```

### Name and Value in form submit

The name attribute of the RadioButton is used to group RadioButton. When the RadioButton are grouped in form, the checked items value attribute will be post to server on form submit that can be retrieved through the name. The disabled RadioButton value will not be sent to the server on form submit.

In the following code snippet, Credit and Debit card is in checked state. Now, the value that is in checked state will be sent on form submit.

#### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Credit/Debit
Card").Name("payment").Checked(true).Render()
@Html.EJS().RadioButton("radio2").Label("Net
Banking").Name("payment").Render()
@Html.EJS().RadioButton("radio3").Label("Cash on
Delivery").Name("payment").Render()
@Html.EJS().RadioButton("radio4").Label("Others").Name("payment").Render()
@Html.EJS().Button("primarybtn").Content("Submit").IsPrimary(true).Render()
<style>
.e-radio-wrapper {
 margin-top: 18px;
}
button {
 margin: 20px 0 0 5px;
}
li {
 list-style: none;
}
</style>
```

#### FORM.CS

```
public ActionResult Form()
{
 return View();
}
```

### Right-To-Left

RadioButton component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in RadioButton component.

#### CSHTML

```
@section ControlsSection{

@Html.EJS().RadioButton("radio1").Label("Default").Name("default").EnableRtl
(true).Render()
}
<style>
.e-radio-wrapper {
 margin-top: 18px;
}
li {
 list-style: none;
}
```

```
}
</style>
```

### RTL.CS

```
public ActionResult Rtl()
{
 return View();
}
```

### Set the disabled state

RadioButton component can be enabled/disabled by giving [disabled](#) property. To disable RadioButton component, the `disabled` property can be set as `true`.

### CSHTML

```
@Html.EJS().RadioButton("radio1").Label("Option
1").Name("default").Checked(true).Render()
@Html.EJS().RadioButton("radio2").Label("Option
2").Name("default").Disabled(true).Render()
@Html.EJS().RadioButton("radio3").Label("Option 3").Name("default").Render()
<style>
.e-radio-wrapper {
 margin-top: 18px;
}
button {
 margin: 20px 0 0 5px;
}
li {
 list-style: none;
}
</style>
```

### DISABLED.CS

```
public ActionResult Disabled()
{
 return View();
}
```

## Migration from Essential JS 1

This article describes the API migration process of RadioButton component from Essential JS 1 to Essential JS 2.

### Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Label | **Property:** `text` <br/><br/> `@Html.EJ().RadioButton("radio").Text("RadioButton")` |

**Property:** `label` <br/><br/> `@Html.EJS().RadioButton("radio").Label("RadioButton").Render()` |

| Checked state | **Property:** *checked* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").Checked(true) | **Property:** *checked*  
 <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").Checked(true).Render() |

| Adding custom css class | **Property:** *cssClass* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").CssClass("custom-class") | **Property:**  
*cssClass* <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").CssClass("custom-  
 class").Render() |

| Disabled state | **Property:** *enabled* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").Enabled(false) | **Property:** *disabled*  
 <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").Disabled(true).Render() |

| State persistence | **Property:** *enablePersistence* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").EnablePersistence(true) | **Property:**  
*enablePersistence* <br/><br/>  
 @Html.EJS().RadioButton("radio").Label("RadioButton").EnablePersistence(true).Render() |

| RTL | **Property:** *enableRTL* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").EnableRTL(true) | **Property:** *enableRtl*  
 <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").EnableRtl(true).Render() |

| HTML Attributes | **Property:** *htmlAttributes* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").HtmlAttributes("") | Not applicable |

| Id property | **Property:** *id* <br/><br/> @Html.EJ().RadioButton("radio").Id("sync") | Not applicable  
 |

| Prefix value of Id | **Property:** *idPrefix* <br/><br/> @Html.EJ().RadioButton("radio").IdPrefix("ej") |  
 Not applicable |

| Name attribute | **Property:** *name* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("Male").Name("gender") | **Property:** *name* <br/><br/>  
 @Html.EJS().RadioButton("radio").Label("Male").Name("gender").Render() |

| Value attribute | **Property:** *value* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("Male").Name("gender").Value("male") | **Property:** *value*  
 <br/><br/>  
 @Html.EJS().RadioButton("radio").Label("Male").Name("gender").Value("male").Render() |

| Size | **Property:** *size* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("Small").Size(RadioButtonSize.Small) | **Property:** *size*  
 <br/><br/> @Html.EJS().RadioButton("radio").Label("Small").CssClass("e-small").Render() |

| Validation rules | **Property:** *validationRules* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").ValidationRules(ViewBag.rules) | Not  
 applicable |

| Validation message | **Property:** *validationMessage* <br/><br/>  
 @Html.EJ().RadioButton("radio").Text("RadioButton").ValidationRules(ViewBag.rules).Validatio  
 nMessage(ViewBag.message) | Not applicable |

## Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroy | **Method:** *destroy* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton") <br/> var radiobutton = \$("#radio").data("ejRadioButton");<br/>radiobutton.destroy(); | **Method:** *destroy* <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").Render() <br/> var radiobutton = document.getElementById('radio').ej2\_instances[0]; <br/>radiobutton.destroy(); |

| Disable the RadioButton | **Method:** *disable* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton") <br/> var radiobutton = \$("#radio").data("ejRadioButton");<br/>radiobutton.disable(); | Not applicable |

| Enable the RadioButton | **Method:** *enable* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton") <br/> var radiobutton = \$("#radio").data("ejRadioButton");<br/>radiobutton.enable(); | Not applicable |

## Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| BeforeChange Event | **Event:** *beforeChange* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton").BeforeChange("beforeChange") <br/><br/>function beforeChange(args) {<br/> &#160;&#160;&#160;&#160;/ code block / <br/>} | Not applicable |

| Change Event | **Event:** *change* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton").Change("change") <br/><br/>function change(args) {<br/> &#160;&#160;&#160;&#160;/ code block / <br/>} | Event: *change* <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").Change("change").Render() <br/><br/>function change(args) {<br/> &#160;&#160;&#160;&#160;/ code block / <br/>} |

| Create Event | **Event:** *create* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton").Create("create") <br/><br/>function create(args) {<br/> &#160;&#160;&#160;&#160;/ code block / <br/>} | Event: *created* <br/><br/> @Html.EJS().RadioButton("radio").Label("RadioButton").Created("created").Render() <br/><br/>function created() {<br/> &#160;&#160;&#160;&#160;/ code block / <br/>} |

| Destroy Event | **Event:** *destroy* <br/><br/> @Html.EJ().RadioButton("radio").Text("RadioButton").Destroy("destroy") <br/><br/>function destroy(args) {<br/> &#160;&#160;&#160;&#160;/ code block / <br/>} | Not applicable |

## Range Navigator

### Getting Started with ASP.NET MVC Range Navigator Control

This section briefly explains about how to include [ASP.NET MVC Range Navigator](#) control in your ASP.NET MVC application using Visual Studio.

### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### **~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.



**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

## Add ASP.NET MVC Range Navigator control

Now, add the Syncfusion ASP.NET MVC Range Navigator control in `~/Views/Home/Index.cshtml` page.

**CSHTML**

```
@(Html.EJS().RangeNavigator("container").Render())
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Range Navigator control will be rendered in the default web browser.



## Populate range navigator with data

Now, we are going to provide data to the range navigator. Add a series object to the range navigator by using series property. Now map the field names x and y in the JSON data to the [xName](#) and [yName](#) properties of the [series](#), then set the JSON data to dataSource property. Since the JSON contains Datetime data, set the [valueType](#) as `DateTime`. By default, the axis valueType is Numeric.

**CSHTML**

```
@model List<RangeNavigatorSample.Controllers.data>
@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(Model).Add();
 }).Render()
)
```

**HOMECONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 }
 }
}
```

```

new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
};
return View(dataSource);
}
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



**Note:** [View Sample in GitHub.](#)

### Selecting Range

The Range Selector's left and right thumbs are used to indicate the selected range in the large collection of data. A range can be selected in the following ways:

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and the end through the `value` property.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Value(ViewBag.range)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### RANGE.CS

```

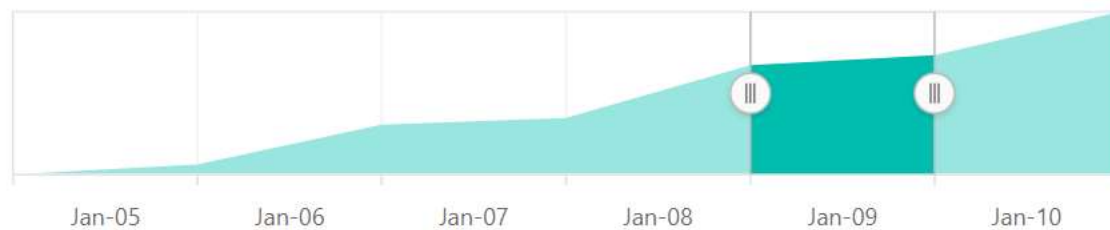
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 }
 }
}

```

```

 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48
 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50
 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66
 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78
 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84
 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### Lightweight range navigator

By default, when the `dataSource` for `series` is empty, a lightweight Range Selector will be shown without Chart.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.Value(ViewBag.range)
.XName("x").YName("y").DataSource(ViewBag.dataSource)
.Render()
)

```

### LIGHT-WEIGHT.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28
 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44
 },
 },
}

```

```

 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48
 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50
 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66
 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78
 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84
 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



## See Also

- [Period Selector](#)

## Series Types

To render the data, the Range Selector supports three types of series.

<!-- markdownlint-disable MD036 -->

### Line

<!-- markdownlint-disable MD036 -->

To render a line series, use series **type** as **Line**. By default, the line series is rendered in the Range Selector.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")

.Tooltip(tl=>tl.Enable(true).Format("yyyy/MM/dd").DisplayMode(Syncfusion.EJ2.Charts.TooltipDisplayMode.Always))

 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Type(Syncfusion.EJ2.Charts.RangeNavigatorType.Line).Add();
 }
)

```

```

 }).Render()
)

```

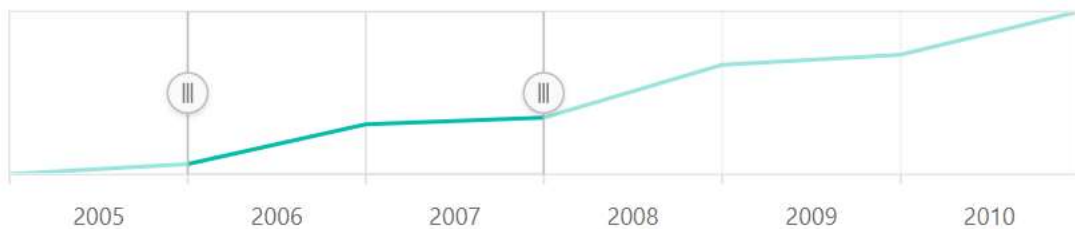
### LINE.CS

```

public IActionResult Index()
{
 List<data>dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### Area

To render an area series, use series **type** as **Area**.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")

.Tooltip(tl=>tl.Enable(true).Format("yyyy/MM/dd").DisplayMode(Syncfusion.EJ2.Charts.TooltipDisplayMode.Always))

```

```

 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Type(Syncfusion.EJ2.Charts.RangeNavigatorType.Area).Add();
 }).Render()
)

```

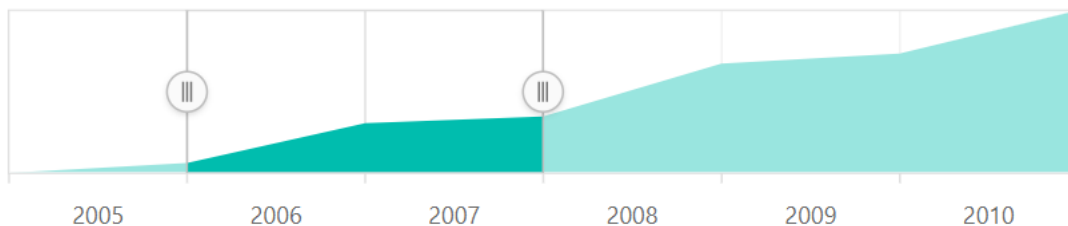
### AREA.CS

```

public IActionResult Index()
{
 List<data>dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### StepLine

To render a Step line series, use series **type** as **Step Line**

### CSHTML

```
@(Html.EJS().RangeNavigator("container"))
```

```

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy").

.Tooltip(tl=>tl.Enable(true).Format("yyyy/MM/dd").DisplayMode(Syncfusion.EJ2.Charts.TooltipDisplayMode.Always))

 .Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Type(Syncfusion.EJ2.Charts.RangeNavigatorType.StepLine).Add();
 }).Render()
)

```

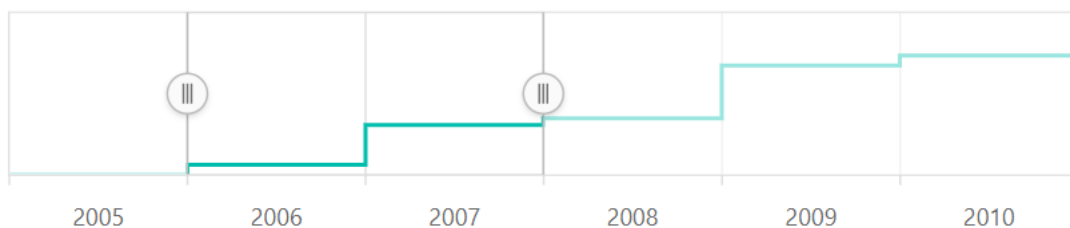
**STEP.CS**

```

public IActionResult Index()
{
 List<data>dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



<!-- markdownlint-disable MD036 -->

## Types of data

### Numeric

The numeric scale is used to represent the numeric values of data in a Range Selector. By default, the **valueType** of a Range Selector is **Double**.

### CSHTML

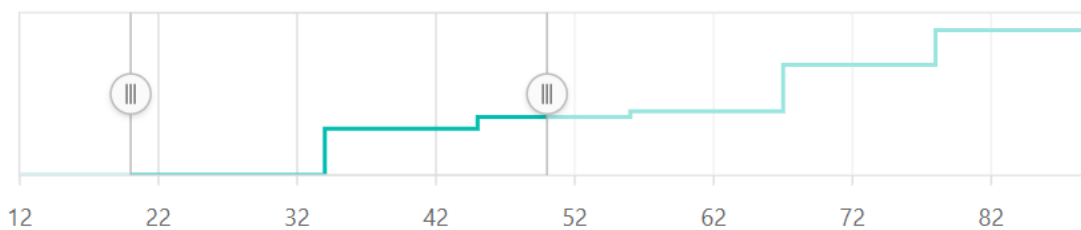
```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

### DOUBLE.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```





### Range

The minimum and the maximum of the scale will be calculated automatically based on the provided data. It can be customized by using the `minimum`, `maximum`, and `interval` properties.

### CSHTML

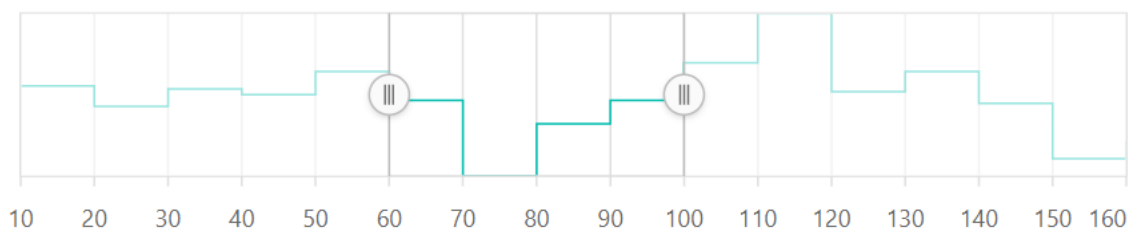
```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .Minimum(10)
 .Maximum(400)
 .Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

### RANGE.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```



### Label Format

The numeric labels can be formatted using the `labelFormat` property and it supports all the globalized formats.

**CSHTML**

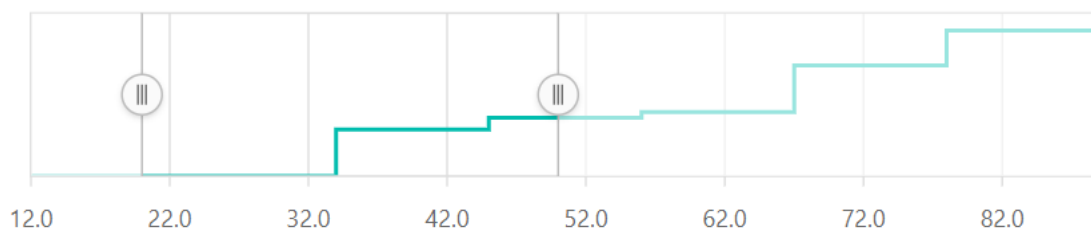
```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

**FORMAT.CS**

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```



The following table describes the result of applying some commonly used label formats on numeric values.

<!-- markdownlint-disable MD033 -->

| Label Value | Label Format property value | Result  | Description                              |
|-------------|-----------------------------|---------|------------------------------------------|
| 1000        | n1                          | 1000.0  | The Number is rounded to 1 decimal place |
| 1000        | n2                          | 1000.00 | The Number is rounded to 2 decimal place |

|      |    |            |                                                                                    |
|------|----|------------|------------------------------------------------------------------------------------|
| 1000 | n3 | 1000.000   | The Number is rounded to 3 decimal place                                           |
| 0.01 | p1 | 1.0%       | The Number is converted to percentage with 1 decimal place                         |
| 0.01 | p2 | 1.00%      | The Number is converted to percentage with 2 decimal place                         |
| 0.01 | p3 | 1.000%     | The Number is converted to percentage with 3 decimal place                         |
| 1000 | c1 | \$1,000.0  | The Currency symbol is appended to number and number is rounded to 1 decimal place |
| 1000 | c2 | \$1,000.00 | The Currency symbol is appended to number and number is rounded to 2 decimal place |

### Custom Label Format

The Range Selector also supports the Custom Label formats using the placeholders such as **{value}\$**, in which the value represents the axis label, e.g. 20\$.

### CSHTML

```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Series(sr =>
 {

sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

### FORMAT.CS

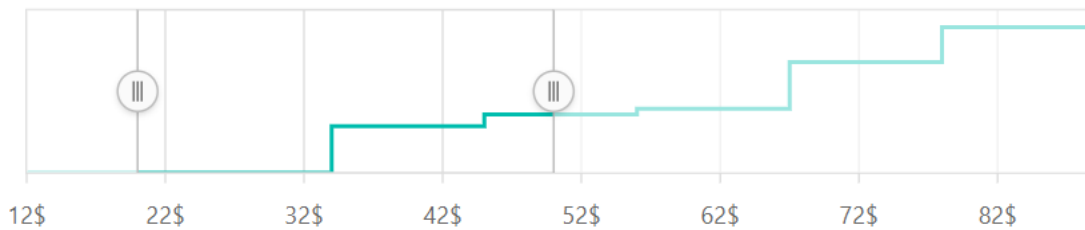
```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
```

```

 public double y1;
 }

```



### Logarithmic Axis

<!-- markdownlint-disable MD033 -->

The Logarithmic supports the logarithmic scale, and it is used to visualize the data when the Range Selector has numerical values in both the lower (e.g.: 10-6) and the higher (e.g.: 106) orders of the magnitude.

### CSHTML

```

@ (Html.EJS() .RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.Logarithmic)
 .Series(sr =>
 {
sr.XName("y").YName("y1").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

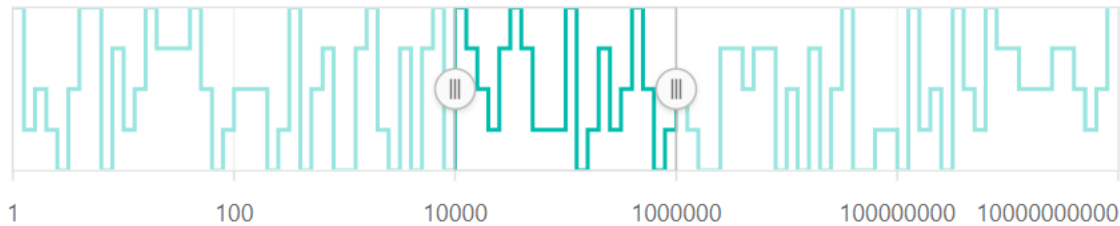
### LOG.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 100, y1 = 44 },
 new data { y = 500, y1 = 48 },
 new data { y = 1000, y1 = 50 },
 new data { y = 5000, y1 = 66 },
 new data { y = 7000, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}

```



### Range

The minimum and the maximum of the Range Selector will be calculated automatically based on the provided data. It can be customized by using the `minimum`, `maximum`, and `interval` properties.

### CSHTML

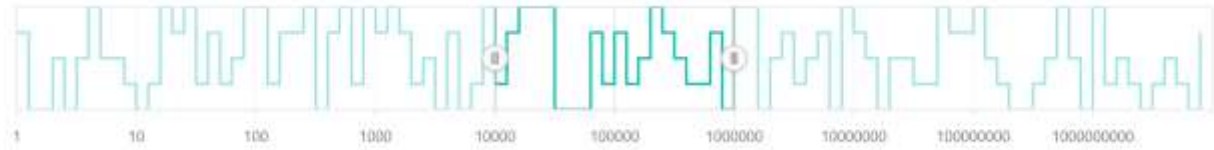
```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.Logarithmic)
 .Minimum(10)
 .Maximum(1000)
 .Series(sr =>
 {
sr.XName("y").YName("y1").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

### LOG-RANGE.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 100, y1 = 44 },
 new data { y = 500, y1 = 48 },
 new data { y = 1000, y1 = 50 },
 new data { y = 5000, y1 = 66 },
 new data { y = 7000, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```



### Logarithmic Base

The Logarithmic Base can be customized using the `logBase` property. The default value of this property is **10**.

### CSHTML

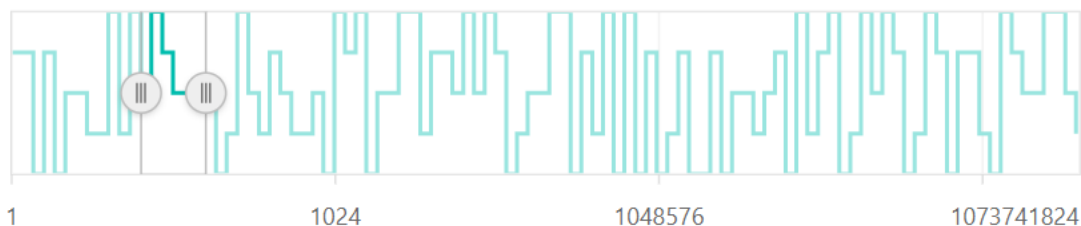
```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.Logarithmic)
 .LogBase(10)
 .Series(sr =>
 {
sr.XName("y").YName("y1").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

### LOG-BASE.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 100, y1 = 44 },
 new data { y = 500, y1 = 48 },
 new data { y = 1000, y1 = 50 },
 new data { y = 5000, y1 = 66 },
 new data { y = 7000, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```



### Date-time

The Range Selector supports the DateTime scale and displays the DateTime values as labels in the specified format.

#### CSHTML

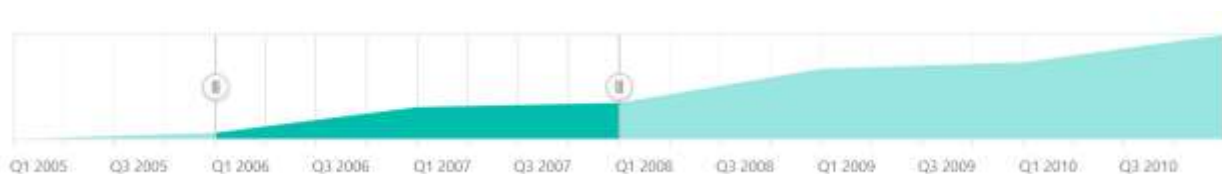
```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

#### DATE-TIME.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}
```



**Note:** Date-time Range navigator supports date-time scale and displays date-time values as a labels in the specified format.

#### Interval Customization

The DateTime intervals can be customized using the `interval` and the `intervalType` properties of the Range Selector. For example, if the `interval` is set to 2 and the `intervalType` is set to years, the interval will be considered to be 2 years.

DateTime supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes

#### CSHTML

```
@(Html.EJS().RangeNavigator("container")
.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.IntervalType("Months")
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

#### DATE-TIME-INTERVAL.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
}
```



```

 ViewBag.dataSource = dataSource;
 return View();
 }
 public class data
 {
 public DateTime x;
 public double y;
 public double y1;
 }

```



### Label Format

The `labelFormat` property is used to format and parse the date to all globalize format.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("'y/M/d'")
 .Series(sr =>
 {
 sr.XName("x").YName("y").Type(Syncfusion.EJ2.Charts.RangeNavigatorType.Area)
 .DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### DATE-TIME-FORMAT.CS

```

public ActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

```

```

 }
 public class data
 {
 public DateTime x;
 public double y;
 public double y1;
 }

```

The following table shows the results of applying some common DateTime formats to the `labelFormat` property.

<!-- markdownlint-disable MD033 -->

| Label Value            | Label Format Property Value | Result      | Description                                            |
|------------------------|-----------------------------|-------------|--------------------------------------------------------|
| new Date(2000, 03, 10) | EEEE                        | Monday      | The Date is displayed in day format                    |
| new Date(2000, 03, 10) | yMd                         | 04/10/2000  | The Date is displayed in month/date/year format        |
| new Date(2000, 03, 10) | MMM                         | Apr         | The Shorthand month for the date is displayed          |
| new Date(2000, 03, 10) | hm                          | 12:00 AM    | Time of the date value is displayed as label           |
| new Date(2000, 03, 10) | hms                         | 12:00:00 AM | The Label is displayed in hours:minutes:seconds format |

## Period selector

The period selector allows to select a range with specified periods.

### Periods

An array of objects that allows the users to specify pre-defined time intervals. The `interval` property specifies the count value of the button, and the `text` property specifies the text to be displayed on the button. The `intervaltype` property allows the users to customize the interval type, and it supports the following types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

### CSHTML

```
@(Html.EJS().RangeNavigator("container"))
```

```

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .PeriodSelectorSettings(ps =>
ps.Position(Syncfusion.EJ2.Charts.PeriodSelectorPosition.Top).Periods(ViewBag.periods))
 .Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### PERIODS.CS

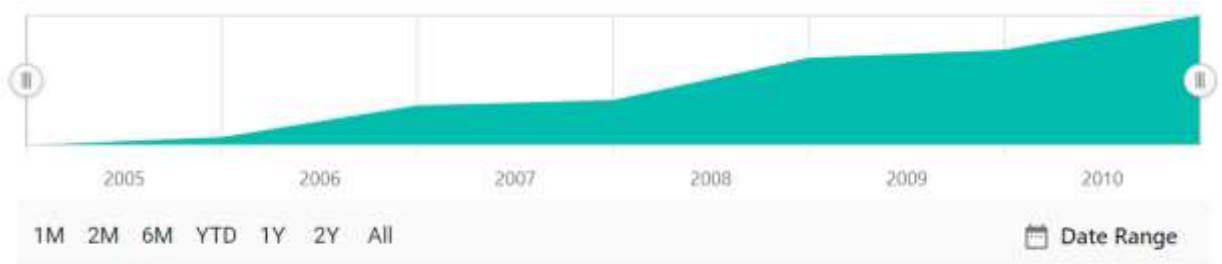
```

public IActionResult Index()
{
 List<data>dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;

 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



## Positioning period selector

The `position` property allows the users to position the period selector at the **Top** or **Bottom**.

**CSHTML**

```
@(Html.EJS().RangeNavigator("container")

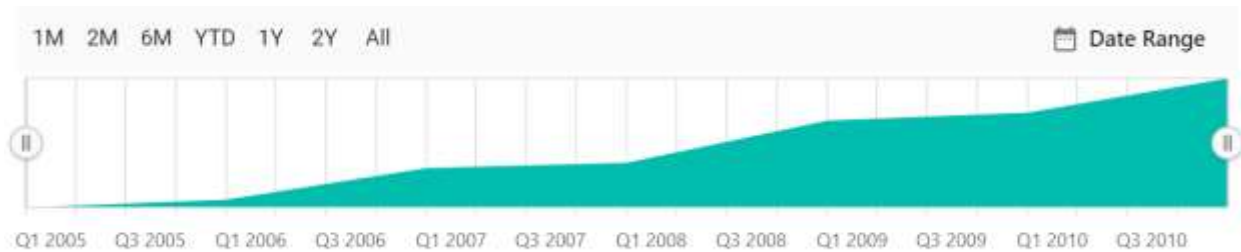
.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.PeriodSelectorSettings(ps =>
ps.Position(Syncfusion.EJ2.Charts.PeriodSelectorPosition.Top).Periods(ViewBag.periods).Position("Bottom"))
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

**POSITION.CS**

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 }
 };
 ViewBag.dataSource = dataSource;

 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}
```



## Height

The **height** property allows the users to specify the height of the period selector. The default value of the height property is **43px**.

## C#HTML

```
@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .PeriodSelectorSettings(ps =>
ps.Position(Syncfusion.EJ2.Charts.PeriodSelectorPosition.Top).Periods(ViewBag.periods).Height("45")
).Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

## HEIGHT.CS

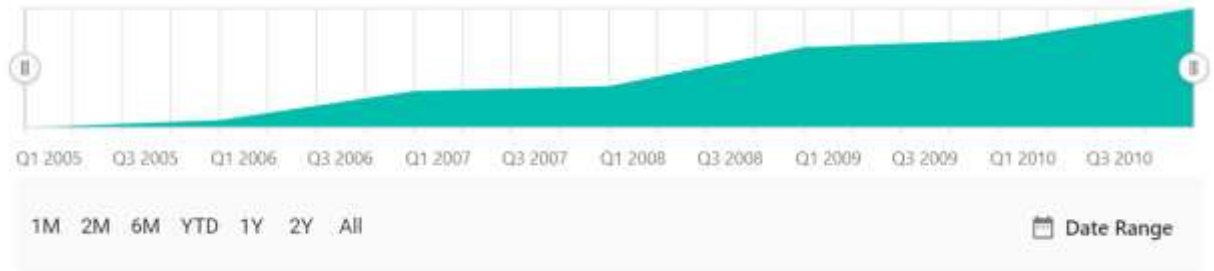
```
public IActionResult Index()
{
 List<data>dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;

 return View();
}
public class data
{
}
```

```

public DateTime x;
public double y;
public double y1;
}

```



### Visibility of range navigator

The `disableRangeSelector` property allows the users to display only the period selector and not the Range Selector.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .DisableRangeSelector("true")
 .PeriodSelectorSettings(ps =>
ps.Position(Syncfusion.EJ2.Charts.PeriodSelectorPosition.Top).Periods(ViewBag.periods))
 .Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### VISIBLE.CS

```

public IActionResult Index()
{
 List<data>dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 }
 },
}

```

```

 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84
 },
 };
 ViewBag.dataSource = dataSource;

 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```

### See Also

- [LightWeight](#)

## Labels

### Multilevel labels

The multi-level labels for the Range Selector can be enabled by setting the `enableGrouping` property to **true**. This is restricted to the DateTime axis alone.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .EnableGrouping(true)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### MULTI.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28
 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44
 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48
 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50
 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66
 },
 };
}

```

```

 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78
 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84
 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### Grouping

The multi-level labels can be grouped using the `groupBy` property with the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy").RangeIntervalType.Months")
 .EnableGrouping(True)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### GROUP.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>

```



```

 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28
 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44
 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48
 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50
 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66
 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78
 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84
 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### Smart labels

The `labelIntersectAction` property is used to avoid overlapping of labels. The following code sample shows the setting of `labelIntersectAction` property to **Hide**.

#### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .EdgeLabelPlacement(Syncfusion.EJ2.Chart.EdgeLabelPlacement.Shift)
 .LabelIntersectAction(Syncfusion.EJ2.Chart.LabelIntersectAction.Hide)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

#### SMART.CS

```

public IActionResult Index()

```

```

{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### Label positioning

By default, the labels can be placed outside the Range Selector. It can also be placed inside the Range Selector using the `labelPosition` property.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### POSITION.CS

```

public IActionResult Index()

```

```

 {
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 },
 };
 ViewBag.dataSource = dataSource;
 return View();
 }
 public class data
 {
 public DateTime x;
 public double y;
 public double y1;
 }

```



### Labels customization

The font size, color, family, etc. can be customized using the `labelStyle` setting.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### CUSTOM.CS

```

public IActionResult Index()
{

```

```

List<data> dataSource = new List<data>
{
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
};
ViewBag.dataSource = dataSource;
return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



## Grid and Tick Lines

### Grid line customization

The gridlines indicate axis divisions by drawing the chart plot. Gridlines include helpful cues to the user, particularly for large or complicated charts. The **width**, **color**, and **dashArray** of the major gridlines can be customized by using the **majorGridLines** setting.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .MajorGridLines(ViewBag.lineWidth)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### GRID.CS

```

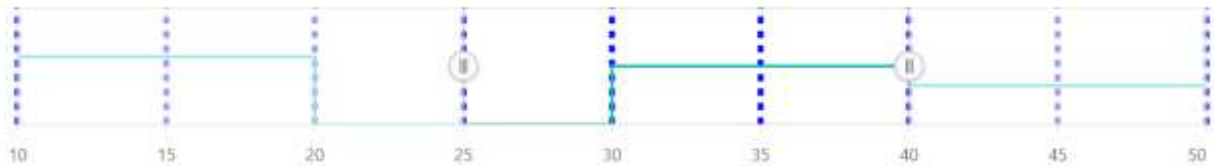
public IActionResult Index()

```

```

{
 List < data>dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public double y;
 public double y1;
}

```



### Tick line customization

Ticklines are the small lines which is drawn on the axis line representing the axis labels. Ticklines will be drawn outside the axis by default. The **width**, **color**, and **dashArray** of the major ticklines can be customized by using the **majorTickLines** setting.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .MajorTickLines(ViewBag.lineWidth)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### TICK.CS

```

public IActionResult Index()
{
 List < data>dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 };
}

```

```

 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public double y;
 public double y1;
}

```



## Customization

### Navigator appearance

The Range Selector can be customized by using the `navigatorStyleSettings`. The `selectedRegionColor` property is used to specify the color for the selected region, whereas the `unselectedRegionColor` property is used to specify the color for the unselected region.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .NavigatorStyleSettings(style => {
style.unselectedRegionColor("Skyblue").selectedRegionColor("black")
 })
 .Series(sr =>
 {
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### APPEARANCE.CS

```

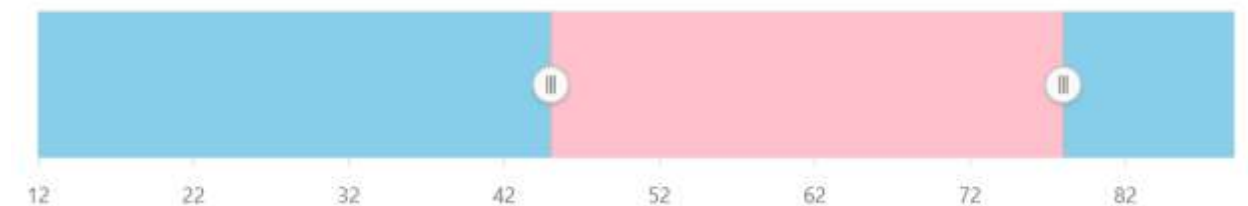
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 }
}

```

```

 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public double y;
 public double y1;
}

```



### Thumb

The thumb property allows to customize the border, fill color, size, and type of thumb. Thumbs can be of two shapes: **Circle** and **Rectangle**.

### CSHTML

```

@ (Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("{value}K")
.NavigatorStyleSettings(style => {

style.unselectedRegionColor("Skyblue").selectedRegionColor("black").Thumb("V
iewBag.thumb)

})
.Series(sr =>
{

sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)

```

### THUMB.CS

```

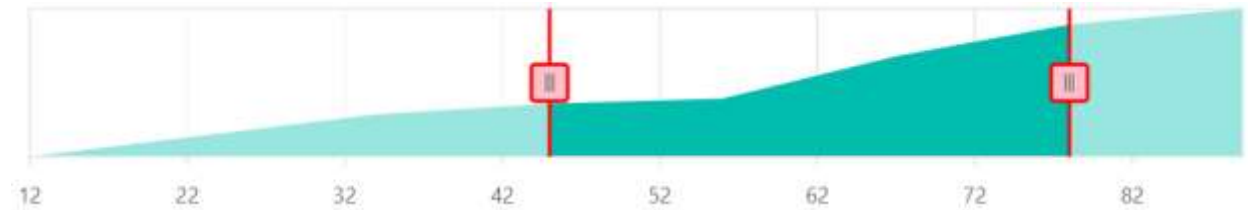
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 }
}

```

```

 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public double y;
 public double y1;
}

```



### Border customization

Using the `navigatorBorder`, the `width` and `color` of the Range Selector border can be customized.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .NavigatorBorder("ViewBag.border")
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

### BORDER.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public double y;
}

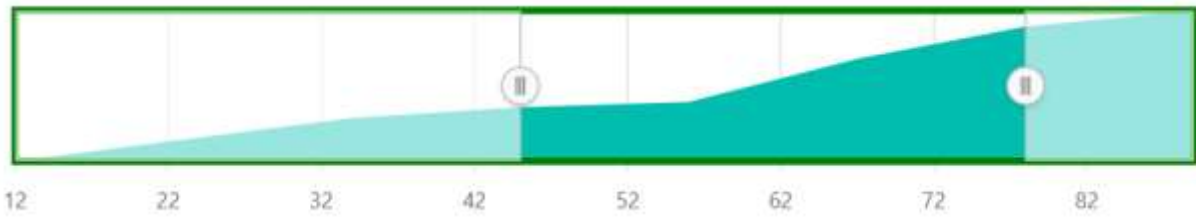
```



```

 public double y1;
 }

```



### Deferred update

If the `enableDeferredUpdate` property is set to **true**, then the changed event will be triggered after dragging the slider. If the `enableDeferredUpdate` is **false**, then the changed event will be triggered when dragging the slider. By default, the `enableDeferredUpdate` is set to **false**.

### CSHTML

```

@ (Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .EnableDeferredUpdate(true)
)
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)

```

### SNAP.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}

```

### Allow snapping

The `allowSnapping` property toggles the placement of the slider exactly to the left or on the nearest interval.

#### CSHTML

```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .AllowSnapping(true)
)
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

#### SNAP.CS

```
public IActionResult Data()
{
 List dataSource = new List
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = chartData;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```

### Animation

The speed of the animation can be controlled using the `animationDuration` property. The default value of the `animationDuration` property is **500** milliseconds.

#### CSHTML

```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("{value}K")
 .AnimationDuration("2000")
)
.Series(sr =>
{
```

```
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
}
```

## ANIMATION.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { y = 12, y1 = 28 },
 new data { y = 34, y1 = 44 },
 new data { y = 45, y1 = 48 },
 new data { y = 56, y1 = 50 },
 new data { y = 67, y1 = 66 },
 new data { y = 78, y1 = 78 },
 new data { y = 89, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public double y;
 public double y1;
}
```

See Also

- [Grid and Tick Lines](#)
- [Labels](#)

## Tooltip

<!-- markdownlint-disable MD036 -->

The tooltip for sliders are supported by the Range Selector. Sliders are used in the Range Selector to select data from a specific range. The tooltip displays the selected start and end values.

<!-- markdownlint-disable MD013 -->

### Enable Tooltip

The tooltip can be used to display information about the selected data and it is enabled by setting the `enable` property to **true**.

## CSHTML

```
@(Html.EJS().RangeNavigator("container")

.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
```

```

.Tooltip(tl=>tl.Enable(true).DisplayMode(Syncfusion.EJ2.Charts.TooltipDisplayMode.Always))
 .Series(sr =>
 {
sr.XName("x").YName("y").Type(Syncfusion.EJ2.Charts.RangeNavigatorType.Area)
.DataSource(ViewBag.dataSource).Add();
 }).Render()
)

```

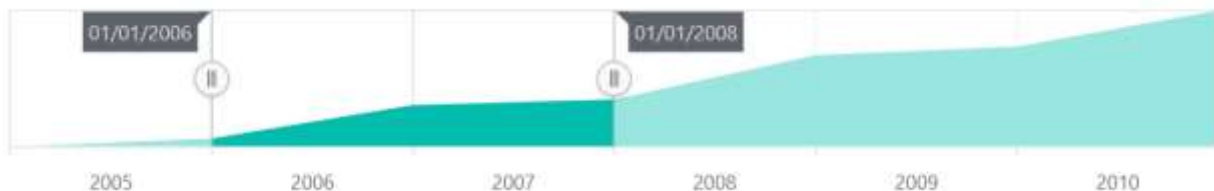
### TOOLTIP.CS

```

public ActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



### Customization

Tooltip can be customized using the following properties:

- enable - Customizes the visibility of the tooltip.
- fill - Customizes the background color of the tooltip.

- opacity - Customizes the opacity of the tooltip.
- textStyle - Customizes the font size, color, family, style, weight, alignment, and overflow of the tooltip.

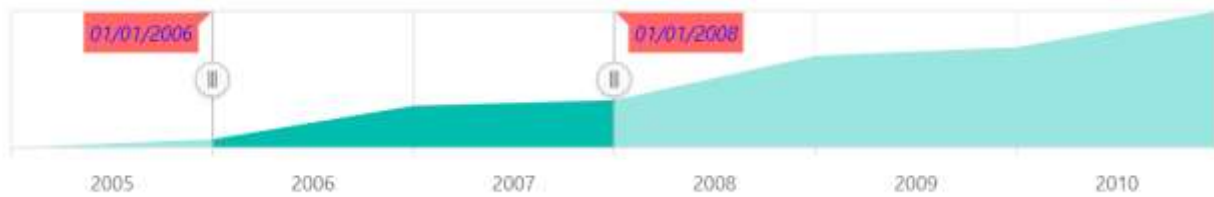
### C#HTML

```
@(Html.EJS().RangeNavigator("container")
.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.Tooltip(tl=>tl.Enable(true).Format("yyyy/MM/dd"))
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

### TOOLTIP.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}
```



### Label Format

The `labelFormat` property in the tooltip is used to format and parse the date to all globalize formats.

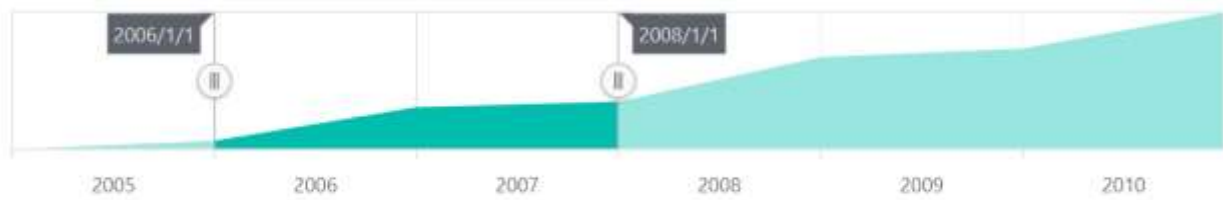
### CSHTML

```
@(Html.EJS().RangeNavigator("container")
.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.Tooltip(tl=>tl.Enable(true).Format("yyyy/MM/dd").DisplayMode(Syncfusion.EJ2
.Charts.TooltipDisplayMode.Always))
.Series(sr =>
{
sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
}).Render()
)
```

### TOOLTIP.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}
```



The following table shows the results of applying some common date and time formats to the `labelFormat` property.

<!-- markdownlint-disable MD033 -->

| Label Value                         | Label Format Property Value | Result      | Description                                            |
|-------------------------------------|-----------------------------|-------------|--------------------------------------------------------|
| <code>new Date(2000, 03, 10)</code> | <code>EEEE</code>           | Monday      | The Date is displayed in day format                    |
| <code>new Date(2000, 03, 10)</code> | <code>yMd</code>            | 04/10/2000  | The Date is displayed in month/date/year format        |
| <code>new Date(2000, 03, 10)</code> | <code>MMM</code>            | Apr         | The Shorthand month for the date is displayed          |
| <code>new Date(2000, 03, 10)</code> | <code>hm</code>             | 12:00 AM    | Time of the date value is displayed as label           |
| <code>new Date(2000, 03, 10)</code> | <code>hms</code>            | 12:00:00 AM | The Label is displayed in hours:minutes:seconds format |

## RTL

The Range Selector supports right-to-left (RTL), which can be enabled with the `enableRtl` property.

### CSHTML

```
@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .EnableRtl(true)
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
```

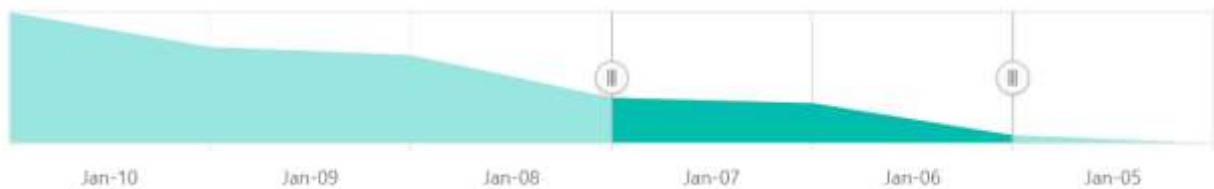
### RTL.CS

```
public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
```

```

 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28
 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44
 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48
 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50
 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66
 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78
 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84
 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```



## Export and print

### Export

The rendered Range Selector can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format by using the **export** method in the Range Selector. This method contains the following parameters:

- **Type** - To specify the export type. The component can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format.
- **File name** - To specify the file name to export.
- **Orientation** - To specify the orientation type. This is applicable only for PDF export type.

### CSHTML

```

@(Html.EJS().RangeNavigator("container")
 .ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
 .LabelFormat("MMM-yy")
 .Series(sr =>
 {
 sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }
)

```



```

 }).Render()
)
 @Html.EJS().Button("export").Content("Primary").IsPrimary(true).Render();
<script>
document.getElementById('export').onclick = function () {
 var control =
document.getElementById('export').ej2_instances[0];
 control.export("PNG", "range");
};
</script>

```

### EXPORT.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}
public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```

### Print

The rendered Range Selector can be printed directly from the browser by calling the public method `print`.

### CSHTML

```

@ (Html.EJS().RangeNavigator("container")
.ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)
.LabelFormat("MMM-yy")
.Series(sr =>
{

```

```

sr.XName("x").YName("y").DataSource(ViewBag.dataSource).Add();
 }).Render()
)
@Html.EJS().Button("print").Content("Primary").IsPrimary(true).Render();
<script>
document.getElementById('print').onclick = function () {
 var chart = document.getElementById('print-
container').ej2_instances[0];
 chart.print();
};
</script>

```

## PRINT.CS

```

public IActionResult Index()
{
 List<data> dataSource = new List<data>
 {
 new data { x = new DateTime(2005, 01, 01), y = 21, y1 = 28 },
 new data { x = new DateTime(2006, 01, 01), y = 24, y1 = 44 },
 new data { x = new DateTime(2007, 01, 01), y = 36, y1 = 48 },
 new data { x = new DateTime(2008, 01, 01), y = 38, y1 = 50 },
 new data { x = new DateTime(2009, 01, 01), y = 54, y1 = 66 },
 new data { x = new DateTime(2010, 01, 01), y = 57, y1 = 78 },
 new data { x = new DateTime(2011, 01, 01), y = 70, y1 = 84 },
 };
 ViewBag.dataSource = dataSource;
 return View();
}

public class data
{
 public DateTime x;
 public double y;
 public double y1;
}

```


## Accessibility in ASP.NET MVC Range navigator component

The Range navigator component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Range navigator component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

```
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

### WAI-ARIA attributes

The Range navigator component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Range navigator component:

- region (role)
- aria-label (attribute)

### Keyboard interaction

The Range navigator component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Range navigator component.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | **Moves the focus to the Range navigator element.** |

| **Ctrl + P** | **Prints the Range navigator.** |

### Ensuring accessibility

The Range navigator component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Range navigator component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Range navigator component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Migration from Essential JS 1

This article describes the API migration process of Range navigator component from Essential JS 1 to Essential JS 2.

#### RangeNavigator

<!-- markdownlint-disable MD033 -->

| Behaviour                  | API in Essential JS 1                                                                            | API in Essential JS 2                                                                                              |
|----------------------------|--------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| Allow snapping             | Property:AllowSnapping@ (Html.EJ().RangeNavigator("container").AllowSnapping(true))              | Property:AllowSnapping@ (Html.EJS().RangeNavigator("container").AllowSnapping(true))                               |
| Animation duration         | Not Applicable                                                                                   | Property:AnimationDuration@ (Html.EJS().RangeNavigator("container").AnimationDuration("3000"))                     |
| Border for range navigator | Property:Border@ (Html.EJS().RangeNavigator("container").Border(b => b.Color("red").Width("2"))) | Property:NavigatorBorder@ (Html.EJS().RangeNavigator("container").NavigatorBorder(b => b.Color("red").Width("2"))) |

|                                              |                                                                                                                                            |                                                                                                                |
|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| data source for range navigator              | <code>Property:DataSource@(Html.EJ().RangeNavigator("container").DataSource())</code>                                                      | <code>Property:DataSource@(Html.EJS().RangeNavigator("container").DataSource(ViewBag.dataSource))</code>       |
| enabling deferred update for range navigator | <code>Property:EnableDeferredUpdate@(Html.EJ().RangeNavigator("container").EnableDeferredUpdate(true))</code>                              | <code>Property:EnableDeferredUpdate@(Html.EJS().RangeNavigator("container").EnableDeferredUpdate(true))</code> |
| multiple level labels                        | <code>Property:LabelSettings.HigherLevelLabels@(Html.EJ().RangeNavigator("container").LabelSettings(l =&gt; l.HigherLevelLabels()))</code> | <code>Property:EnableGrouping@(Html.EJS().RangeNavigator("container").EnableGrouping(false))</code>            |
| enabling scrollbar                           | <code>Property:EnableScrollBar@(Html.EJ().RangeNavigator("container").EnableScrollBar(true))</code>                                        | Not Applicable                                                                                                 |
| enabling auto resizing                       | <code>Property:EnableAutoResize@(Html.EJ().RangeNavigator("container").EnableAutoResize(true))</code>                                      | Not Applicable                                                                                                 |

|                                                                   |                                                                                                                                                      |                                                                                                                             |
|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| res<br>pon<br>sive<br>of<br>ran<br>ge<br>nav<br>igat<br>or        | Property:IsResponsive@ (Html.EJ().RangeNavigator("container").IsResponsive(true))                                                                    | Not Applicable                                                                                                              |
| ena<br>blin<br>g<br>RTL<br>for<br>ran<br>ge<br>nav<br>igat<br>or  | Property:EnableRtl@ (Html.EJ().RangeNavigator("container").EnableRtl(true))                                                                          | Property:EnableRtl@ (Html.EJS().RangeNavigator("container").EnableRtl(true))                                                |
| inte<br>rval<br>for<br>ran<br>ge<br>nav<br>igat<br>or             | Property:ValueAxisSettings.Range.Interval@ (Html.EJ().RangeNavigator("container").ValueAxisSettings(v => v.Range(r => r.Interval("1"))))             | Property:Interval@ (Html.EJS().RangeNavigator("container").Interval("1"))                                                   |
| inte<br>rval<br>typ<br>e<br>for<br>ran<br>ge<br>nav<br>igat<br>or | Property:ValueAxisSettings.Range.IntervalType@ (Html.EJ().RangeNavigator("container").ValueAxisSettings(v => v.Range(r => r.IntervalType("Years")))) | Property:IntervalType@ (Html.EJS().RangeNavigator("container").IntervalType(Syncfusion.EJ2.Charts.RangeIntervalType.Years)) |
| lab<br>elfo<br>rma<br>t<br>for<br>ran<br>ge<br>nav                | Not applicable                                                                                                                                       | Property:LabelFormat@ (Html.EJS().RangeNavigator("container").LabelFormat(yMd))                                             |

|                                                                                       |                                                                                                                                                                                     |                                                                                                                                                                     |
|---------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| igat<br>or                                                                            |                                                                                                                                                                                     |                                                                                                                                                                     |
| lab<br>el<br>inte<br>rsec<br>t<br>acti<br>on<br>for<br>ran<br>ge<br>nav<br>igat<br>or | Not applicable                                                                                                                                                                      | Property:LabelIntersectAction@ (Html.EJS () .RangeNav<br>igator ("container") .LabelIntersectAction (Syn<br>cfusion.EJ2.Charts.RangeLabelIntersectAction<br>.Hide)) |
| lab<br>elSt<br>yle<br>ran<br>ge<br>nav<br>igat<br>or                                  | Property:ValueAxisSettings.Font@ (Htm<br>l.EJ () .RangeNavigator ("contai<br>ner") .ValueAxisSettings (v =><br>v.Font ()))                                                          | Property:LabelStyle@ (Html.EJS () .RangeNavigator ("<br>container") .LabelStyle ())                                                                                 |
| loca<br>le<br>of<br>ran<br>ge<br>nav<br>igat<br>or                                    | Property:Locale@ (Html.EJ () .RangeN<br>avigator ("container") .Locale ("<br>en-US"))                                                                                               | Property:Locale@ (Html.EJS () .RangeNavigator ("con<br>tainer") .Locale ("en-US"))                                                                                  |
| maj<br>or<br>grid<br>line<br>s of<br>ran<br>ge<br>nav<br>igat<br>or                   | Property:ValueAxisSettings.MajorGridLi<br>nes@ (Html.EJ () .RangeNavigator ("<br>container") .ValueAxisSetting<br>s (v => v.MajorGridLines (m =><br>m.Width ("2") .Color ("red")))) | Property:MajorGridLines@ (Html.EJS () .RangeNavigat<br>or ("container") .MajorGridLines (mg =><br>mg.Color ("blue") .Width (2) .DashArray ("5,5")))                 |
| mar<br>gin<br>of                                                                      | Not Applicable                                                                                                                                                                      | Property:Margin@ (Html.EJS () .RangeNavigator ("co<br>ntainer") .Margin ())                                                                                         |

|                                          |                                                                                                                                                   |                                                                                                                  |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| range navigator                          |                                                                                                                                                   |                                                                                                                  |
| maximum value of range navigator         | <b>Property:ValueAxisSettings.Range.Maximum@</b><br>(Html.EJ().RangeNavigator("container").ValueAxisSettings(v => v.Range(r => r.Maximum("34")))) | <b>Property:Maximum@</b> (Html.EJS().RangeNavigator("container").Maximum("34"))                                  |
| minimum value of range navigator         | <b>Property:ValueAxisSettings.Range.Minimum@</b><br>(Html.EJ().RangeNavigator("container").ValueAxisSettings(v => v.Range(r => r.Minimum("10")))) | <b>Property:Minimum@</b> (Html.EJS().RangeNavigator("container").Minimum("10"))                                  |
| query for data source of range navigator | Not Applicable                                                                                                                                    | <b>Property:Query@</b> (Html.EJS().RangeNavigator("container").Query())                                          |
| Secondary label                          | Not Applicable                                                                                                                                    | <b>Property:SecondaryLabelAlignment@</b> (Html.EJS().RangeNavigator("container").SecondaryLabelAlignment("Far")) |



|                                                                      |                                                                                                                                                                         |                                                                                                              |
|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| align<br>ment<br>of<br>range<br>nav<br>igat<br>or                    |                                                                                                                                                                         |                                                                                                              |
| Skel<br>eto<br>n of<br>range<br>nav<br>igat<br>or<br>axis            | Not Applicable                                                                                                                                                          | <b>Property:</b> <code>Skeleton@ (Html.EJS () .RangeNavigator ("c<br/>ontainer") .Skeleton ())</code>        |
| skel<br>eto<br>nty<br>pe<br>of<br>range<br>nav<br>igat<br>or<br>axis | Not Applicable                                                                                                                                                          | <b>Property:</b> <code>SkeletonType@ (Html.EJS () .RangeNavigator<br/>("container") .SkeletonType ())</code> |
| The<br>me<br>of<br>range<br>nav<br>igat<br>or                        | <b>Property:</b> <code>Theme@ (Html.EJS () .Rang<br/>eNavigator ("container") .Theme<br/>())</code>                                                                     | <b>Property:</b> <code>Theme@ (Html.EJS () .RangeNavigator ("co<br/>ntainer") .Theme ())</code>              |
| Def<br>ault<br>sele<br>ctor<br>val<br>ue<br>ran<br>ge                | <b>Property:</b> <code>SelectedRangeSettings@ (Html<br/>.EJ () .RangeNavigator ("contain<br/>er") .SelectedRangeSettings (s<br/>=&gt; s.Start ("2") .End ("20"))</code> | <b>Property:</b> <code>Value@ (Html.EJS () .RangeNavigator ("con<br/>tainer") .Value ("2,10"))</code>        |

|                                                                     |                                                                                                             |                                                                                                                       |
|---------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| nav<br>igat<br>or                                                   |                                                                                                             |                                                                                                                       |
| Val<br>ue<br>typ<br>e of<br>ran<br>ge<br>nav<br>igat<br>or          | Property:ValueType\$(Html.EJS().RangeNavigator("container").ValueType("DateTime"))                          | Property:valueType@ (Html.EJS().RangeNavigator("container").ValueType(Syncfusion.EJ2.Charts.RangeValueType.DateTime)) |
| Wid<br>th<br>of<br>ran<br>ge<br>nav<br>igat<br>or                   | Property:Size.Width@ (Html.EJ().RangeNavigator("container").Size(s => s.Width("400")))                      | Property:Width@ (Html.EJS().RangeNavigator("container").Width("400"))                                                 |
| Hei<br>ght<br>of<br>ran<br>ge<br>nav<br>igat<br>or                  | Property:Size.Height@ (Html.EJ().RangeNavigator("container").Size(s => s.Height("400")))                    | Property:Height@ (Html.EJS().RangeNavigator("container").Height("400"))                                               |
| Seri<br>es<br>sett<br>ings<br>for<br>ran<br>ge<br>nav<br>igat<br>or | Property:SeriesSettings@ (Html.EJ().RangeNavigator("container").SeriesSettings())                           | Not Applicable                                                                                                        |
| Ran<br>ge<br>sett<br>ings<br>for<br>ran                             | Property:RangeSettings@ (Html.EJ().RangeNavigator("container").RangeSettings(r => r.Start("20").End("30"))) | Not Applicable                                                                                                        |

|                                                                                  |                                                                                                                                |                |
|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------|
| ge<br>nav<br>igat<br>or                                                          |                                                                                                                                |                |
| Scr<br>oll<br>ran<br>ge<br>sett<br>ings<br>for<br>ran<br>ge<br>nav<br>igat<br>or | Property:ScrollRangeSettings@ (Html.EJ()<br>.RangeNavigator("container").ScrollRangeSettings(r =><br>r.Start("20").End("30"))) | Not Applicable |

## Series

&lt;!-- markdownlint-disable MD033 --&gt;

|                                                                   |                                                                                                        |                                                                                                                     |
|-------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| Beh<br>avio<br>ur                                                 | API in Essential JS 1                                                                                  | API in Essential JS 2                                                                                               |
| ani<br>mat<br>ion                                                 | Property:EnableAnimation@ (Html.EJ()<br>.RangeNavigator("container").EnableAnimation(true))            | Property:Animation@ (Html.EJS().RangeNavigator<br>("container").Animation(a =><br>a.Enable(true).Duration("3000"))) |
| Bor<br>der<br>for<br>ran<br>ge<br>navi<br>gat<br>or<br>seri<br>es | Not Applicable                                                                                         | Property:Border@ (Html.EJS().RangeNavigator("c<br>ontainer").Border(b =><br>b.Color("red").Width("2")))             |
| dat<br>aSo<br>urce<br>for<br>ran<br>ge<br>navi                    | Property:Series.DataSource@ (Html.EJ()<br>.RangeNavigator("container").Series(s =><br>s.DataSource())) | Property:Series.DataSource@ (Html.EJS().RangeNav<br>igator("container").Series(s =><br>s.DataSource()))             |

|                                                                                    |                                                                                                   |                                                                                                                   |
|------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|
| gat<br>or                                                                          |                                                                                                   |                                                                                                                   |
| que<br>ry<br>for<br>dat<br>a<br>sour<br>ce<br>of<br>ran<br>ge<br>navi<br>gat<br>or | Not Applicable                                                                                    | Property:Query@(Html.EJS().RangeNavigator("container").Series(s => s.Query()))                                    |
| seri<br>es<br>typ<br>e<br>for<br>ran<br>ge<br>navi<br>gat<br>or                    | Property:Series.Type@(Html.EJ().RangeNavigator("container").Series(s => s.Type(SeriesType.Line))) | Property:Series.Type@(Html.EJS().RangeNavigator("container").Type(Syncfusion.EJ2.Charts.RangeNavigatorType.Line)) |
| seri<br>es<br>xNa<br>me<br>for<br>ran<br>ge<br>navi<br>gat<br>or                   | Property:Series.XName@(Html.EJ().RangeNavigator("container").Series(s => s.XName()))              | Property:Series.XName@(Html.EJS().RangeNavigator("container").Series(s => s.XName()))                             |
| seri<br>es<br>yNa<br>me<br>for<br>ran<br>ge<br>navi<br>gat<br>or                   | Property:Series.YName@(Html.EJ().RangeNavigator("container").Series(s => s.YName()))              | Property:Series.YName@(Html.EJS().RangeNavigator("container").Series(s => s.YName()))                             |

|                                       |                                                                                                         |                                                                                                                     |
|---------------------------------------|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| series fill color for range navigator | <code>Property:Series.Fill@(Html.EJ().RangeNavigator("container").Series(s =&gt; s.Fill("red")))</code> | <code>Property:Series.Fill@(Html.EJS().RangeNavigator("container").Series(s =&gt; s.Fill("red")))</code>            |
| series width for range navigator      | <code>Property:Series.Width@(Html.EJ().RangeNavigator("container").Series(s =&gt; s.Width("")))</code>  | <code>Property:Series.Width@(Html.EJS().RangeNavigator("container").Series(s =&gt; s.Width("")))</code>             |
| series dashArray for range navigator  | Not Applicable                                                                                          | <code>Property:Series.DashArray@(Html.EJS().RangeNavigator("container").Series(s =&gt; s.DashArray("10,5")))</code> |

### StyleSettings

<!-- markdownlint-disable MD033 -->

| Behaviour                         | API in Essential JS 1                                                                                         | API in Essential JS 2                                                                        |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| Style settings of range navigator | <code>Property:NavigatorStyleSettings@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings())</code> | <code>Property:StyleSettings@(Html.EJS().RangeNavigator("container").StyleSettings())</code> |

|                                                                     |                                                                                                                                        |                                                                                                                            |
|---------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Select<br>ed<br>region<br>color<br>of<br>range<br>naviga<br>tor     | Property:SelectedRegionColor@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.SelectedRegionColor("red")))         | Property:SelectedRegionColor@(Html.EJS().RangeNavigator("container").StyleSettings(s => s.SelectedRegionColor("red")))     |
| UnSe<br>lected<br>region<br>color<br>of<br>range<br>naviga<br>tor   | Property:UnselectedRegionColor@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.UnSelectedRegionColor("red")))     | Property:UnselectedRegionColor@(Html.EJS().RangeNavigator("container").StyleSettings(s => s.UnSelectedRegionColor("red"))) |
| Thumb<br>color<br>of<br>range<br>naviga<br>tor                      | Property:ThumColor@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.ThumColor("red")))                             | Property:ThumbSettings@(Html.EJS().RangeNavigator("container").StyleSettings(s => s.ThumbSettings("red")))                 |
| Select<br>ed<br>region<br>opacity<br>of<br>range<br>naviga<br>tor   | Property:SelectedRegionOpacity@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.SelectedRegionOpacity("0.4")))     | Not Applicable                                                                                                             |
| Unse<br>lected<br>region<br>opacity<br>of<br>range<br>naviga<br>tor | Property:UnselectedRegionOpacity@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.UnSelectedRegionOpacity("0.4"))) | Not Applicable                                                                                                             |
| Backgr<br>ound<br>for<br>thumb                                      | Property:Background@(Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.Background("red")))                           | Not Applicable                                                                                                             |

|                                             |                                                                                                                                |                |
|---------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|----------------|
| border for thumb                            | Property:Border@ (Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.Border(b => b.Color("red").Width("2")))) | Not Applicable |
| Highlight settings for range navigator      | Property:HighlightSettings@ (Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.HighlightSettings()))         | Not Applicable |
| Selected style settings for range navigator | Property:SelectionSettings@ (Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.SelectionSettings()))         | Not Applicable |
| Left thumb template for range navigator     | Property:LeftThumbTemplate@ (Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.LeftThumbTemplate()))         | Not Applicable |
| Right thumb template for range navigator    | Property:RightThumbTemplate@ (Html.EJ().RangeNavigator("container").NavigatorStyleSettings(n => n.RightThumbTemplate()))       | Not Applicable |

### Tooltip

<!-- markdownlint-disable MD033 -->

| Behaviour        | API in Essential JS 1                                                                                    | API in Essential JS 2                                                                  |
|------------------|----------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| tooltip          | Property:Visible@ (Html.EJ().RangeNavigator("container").Tooltip(t => t.Visible(true)))                  | Property:Enable@ (Html.EJS().RangeNavigator("container").Tooltip(t => t.Enable(true))) |
| background color | Property:BackgroundColor@ (Html.EJ().RangeNavigator("container").Tooltip(t => t.BackgroundColor("red"))) | Property:Fill@ (Html.EJ().RangeNavigator("container").Tooltip(t => t.Fill("red")))     |

|                         |                                                                                                                            |                                                                                                                           |
|-------------------------|----------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| of tooltip              |                                                                                                                            |                                                                                                                           |
| Font style of tooltip   | <b>Property:</b> Font@ (Html.EJ() .RangeNavigator("container").Tooltip(t => t.Font()))                                     | <b>Property:</b> TextStyle@ (Html.EJS() .RangeNavigator("container").Tooltip(t => t.TextStyle()))                         |
| Label format of tooltip | <b>Property:</b> LabelFormat@ (Html.EJ() .RangeNavigator("container").Tooltip(t => t.LabelFormat("yMd")))                  | <b>Property:</b> Format@ (Html.EJS() .RangeNavigator("container").Tooltip(t => t.Format("yMd")))                          |
| Display mode of tooltip | <b>Property:</b> TooltipDisplayMode@ (Html.EJ() .RangeNavigator("container").Tooltip(t => t.TooltipDisplayMode("always"))) | <b>Property:</b> displayMode@ (Html.EJS() .RangeNavigator("container").Tooltip(t => t.DisplayMode("always")))             |
| Template of tooltip     | Not Applicable                                                                                                             | <b>Property:</b> Template@ (Html.EJS() .RangeNavigator("container").Tooltip(t => t.Template("")))                         |
| Border of tooltip       | Not Applicable                                                                                                             | <b>Property:</b> Border@ (Html.EJS() .RangeNavigator("container").Tooltip(t => t.Border(b => b.Color("red").Width("2")))) |
| Opacity of tooltip      | Not Applicable                                                                                                             | <b>Property:</b> Opacity@ (Html.EJS() .RangeNavigator("container").Tooltip(t => t.Opacity("0.5")))                        |

### Period Selector

<!-- markdownlint-disable MD033 -->

|                 |                       |                                                                                                                                                                    |
|-----------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Behavior        | API in Essential JS 1 | API in Essential JS 2                                                                                                                                              |
| period selector | Not Applicable        | <b>Property:</b> PeriodSelector (Html.EJS() .RangeNavigator("container").PeriodSelector(p => p.Periods(pr => pr.Interval("1").IntervalType("Months").Text("1M")))) |



## Methods

&lt;!-- markdownlint-disable MD033 --&gt;

| Behavior | API in Essential JS 1 | API in Essential JS 2                                                                                |
|----------|-----------------------|------------------------------------------------------------------------------------------------------|
| Print    | Not Applicable        | <code>Property:Print()@ (Html.EJS () .RangeNavigator ("container") RangeNavigator.Print ())</code>   |
| Export   | Not Applicable        | <code>Property:export()@ (Html.EJS () .RangeNavigator ("container") RangeNavigator.Export ())</code> |

## Events

&lt;!-- markdownlint-disable MD033 --&gt;

| Behavior                                         | API in Essential JS 1                                                                           | API in Essential JS 2                                                                  |
|--------------------------------------------------|-------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| Fires before loading the Range Navigator.        | <code>Property:Load@ (Html.EJ () .RangeNavigator ("container") .Load ())</code>                 | <code>Property:Load@ (Html.EJS () .RangeNavigator ("container") .Load ())</code>       |
| Fires before loading the Range Navigator.        | <code>Property:Loaded@ (Html.EJ () .RangeNavigator ("container") .Loaded ())</code>             | <code>Property:Loaded@ (Html.EJS () .RangeNavigator ("container") .Loaded ())</code>   |
| Fires when the value changes in range navigator. | <code>Property:RangeChanged@ (Html.EJ () .RangeNavigator ("container") .RangeChanged ())</code> | <code>Property:Changed@ (Html.EJS () .RangeNavigator ("container") .Changed ())</code> |

|                                                    |                                                                                                 |                                                                                          |
|----------------------------------------------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| Fires before after resize the Range Navigator.     | Not Applicable                                                                                  | Property:Resized@ (Html.EJS () .RangeNavigator ("container") .Resized () )               |
| Fires before tooltip the Range Navigator.          | Not Applicable                                                                                  | Property:TooltipRender@ (Html.EJS () .RangeNavigator ("container") .TooltipRender () )   |
| Fires before period render in the Range Navigator. | Not Applicable                                                                                  | Property:SelectorRender@ (Html.EJS () .RangeNavigator ("container") .SelectorRender () ) |
| Fires when scrollStart the Range Navigator.        | Property:ScrollStart@ (Html.EJ () .RangeNavigator ("container") .ScrollStart () )               | Not Applicable                                                                           |
| Fires when scrollEnd the Range Navigator.          | Property:ScrollEnd@ (Html.EJ () .RangeNavigator ("container") .ScrollEnd () )                   | Not Applicable                                                                           |
| Fires when selected range Start the                | Property:SelectedRangeStart@ (Html.EJ () .RangeNavigator ("container") .SelectedRangeStart () ) | Not Applicable                                                                           |

|                                                          |                                                                                       |                |
|----------------------------------------------------------|---------------------------------------------------------------------------------------|----------------|
| Range Navigator.                                         |                                                                                       |                |
| Fires when selected range ends the Range Navigator.      | Property:SelectedRangeEnd@ (Html.EJ().RangeNavigator("container").SelectedRangeEnd()) | Not Applicable |
| Fires when scroll range change d in the Range Navigator. | Property:ScrollChanged@ (Html.EJ().RangeNavigator("container").ScrollChanged())       | Not Applicable |
| Fires when click in the Range Navigator.                 | Property:Click@ (Html.EJ().RangeNavigator("container").Click())                       | Not Applicable |
| Fires when right click in the Range Navigator.           | Property:RightClick@ (Html.EJ().RangeNavigator("container").RightClick())             | Not Applicable |
| Fires when double click in the Range                     | Property:DoubleClick@ (Html.EJ().RangeNavigator("container").DoubleClick())           | Not Applicable |

|                |  |  |
|----------------|--|--|
| Naviga<br>tor. |  |  |
|----------------|--|--|

## Range Slider

### Getting Started with ASP.NET MVC Range Slider Control

This section briefly explains about how to include [ASP.NET MVC Range Slider](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

**~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Range Slider control

Now, add the Syncfusion ASP.NET MVC Range Slider control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().Slider("slider").Type(Syncfusion.EJ2.Inputs.SliderType.MinRange)
.Value(40).Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Range Slider control will be rendered in the default web browser.



**Note:** [View Sample in GitHub](#)

See also

- [Slider Types](#)
- [Slider Formatting](#)
- [Orientation Slider](#)
- [Ticks in Slider](#)
- [Tooltip in Slider](#)

## Types in RangeSlider Control

The types of Slider are as follows:

### | Types | Usage |

| --- | --- |

| Default | Shows a default Slider to select a single value. |

| MinRange | Displays the shadow from the start value to the current selected value. |

| Range | Selects a range of values. It also displays the shadow in-between the selection range. |

**Note:** Both the Default Slider and Min-Range Slider have same behavior that is used to select a single value. In Min-Range Slider, a shadow is considered from the start value to current handle position. But the Range Slider contains two handles that is used to select a range of values and a shadow is considered in between the two handles.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
@Html.EJS().Slider("default").Value("30").Render()

@Html.EJS().Slider("minrange").Type(SliderType.MinRange).Value("30").Render()

@Html.EJS().Slider("range").Value(ViewBag.range).Type(SliderType.Range).Render()
```

### HOMECONTROLLER.CS

```
public IActionResult Index()
{
 ViewBag.range = new int[] { 30, 70 };
 return View();
}
```

Default



MinRange



Range



## Tooltip

The Slider displays the tooltip to indicate the current value by clicking the Slider bar or drag the Slider handle. The Tooltip position can be customized by using the `placement` property. Also decides the tooltip display mode on a page, i.e., on hovering, focusing, or clicking on the Slider handle and it always remains/displays on the page.

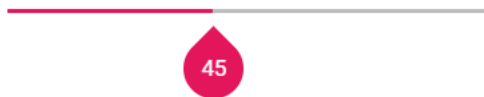
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Type(SliderType.MinRange).Value("30").Tooltip(
new SliderTooltipData { IsVisible = true, ShowOn = TooltipShowOn.Always,
Placement = TooltipPlacement.After }).Render()
```

### HOMECONTROLLER.CS

```
public IActionResult Index()
{
 return View();
}
```



## Buttons

The Slider value can be changed by using the Increase and Decrease buttons. In Range Slider, by default the first handle value will be changed while clicking the button. Change the handle focus and press the button to change the last focused handle value.

**Note:** After enabling the slider buttons if the 'Tab' key is pressed, the focus goes to the handle and not to the button.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").ShowButtons(true).Type(SliderType.MinRange).Value("30").Tooltip(new SliderTooltipData { IsVisible = true, ShowOn =
TooltipShowOn.Always, Placement = TooltipPlacement.After }).Render()
```

### HOMECONTROLLER.CS

```
public IActionResult Index()
{
 return View();
}
```



## Orientation

The Slider can be displayed, either in horizontal or vertical orientation. By default, the Slider renders in horizontal orientation.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Value("30").Orientation(SliderOrientation.Vertical).Render()
<style>
 #default {
 height: 300px;
 }
</style>
```

### HOMECONTROLLER.CS

```
public IActionResult Index()
{
 return View();
}
```



## Ticks

The Ticks in Slider supports you to easily identify the current value/values of the Slider. It contains `smallStep` and `largeStep`. The value of the major ticks alone will be displayed in the slider. In order to enable/disable the small ticks, use the `showSmallTicks` property.

### CSHTML



```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
@Html.EJS().Slider("default").Value("30").Ticks(new SliderTicksData {
Placement = Placement.After, LargeStep = 20, SmallStep = 10, ShowSmallTicks
= true }).Tooltip(new SliderTooltipData { IsVisible = true, Placement =
TooltipPlacement.Before, ShowOn = TooltipShowOn.Always }).Render()
```

### TICKS.CS

```
public ActionResult Ticks()
{
 return View();
}
```



### Step

When the Slider is moved, it increases/decreases the value based on the step value. By default, the value is increased/decreased by 1. Use the `step` property to change the increment step value.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
@Html.EJS().Slider("default").Value("30").Step(10).Ticks(new
SliderTicksData { Placement = Placement.After, LargeStep = 20, SmallStep =
10, ShowSmallTicks = true }).Tooltip(new SliderTooltipData { IsVisible =
true, Placement = TooltipPlacement.Before, ShowOn = TooltipShowOn.Always
}).Render()
```

### STEP.CS

```
public ActionResult Step()
{
 return View();
}
```



### Min and Max

Enables the minimum/starting and maximum/ending value of the Slider, by using the `min` and `max` property. By default, the minimum value is 1 and maximum value is 100. In the following sample the slider is rendered with the min value as 100 and max value as 1000.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(100).Max(1100).Value("300").Tooltip(new
SliderTooltipData { isVisible = true, ShowOn = TooltipShowOn.Always,
Placement = TooltipPlacement.Before }).Ticks(new SliderTicksData { Placement
= Placement.After, LargeStep = 200, SmallStep = 100, ShowSmallTicks = true
}).Render()
```

#### MIN-MAX.CS

```
public ActionResult MiniMax()
{
 return View();
}
```



### Formatting in Range Slider Control

The `format` feature used to customize the units of Slider values to desired format. The formatted values will also be applied to the ARIA attributes of the slider. There are two ways of achieving formatting in slider.

- Use the `format` API of slider which utilizes our [Internationalization](#) to format values.
- Customize using the events namely `renderingTicks` and `tooltipChange`.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(0).Max(100).Value("30").Step(1).Tooltip(new
SliderTooltipData { IsVisible = true, Format = "C2" }).Ticks(new
SliderTicksData { Placement = Placement.After, LargeStep = 20, SmallStep =
10, ShowSmallTicks = true, Format = "C2" }).Render()
```

### FORMAT.CS

```
public ActionResult Format()
{
 return View();
}
```



### Using format API

In this method, we have different predefined formatting styles like Numeric (N), Percentage (P), Currency (C) and # specifiers. In this below example we have formatted the ticks and tooltip values into percentage.

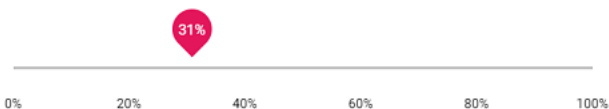
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(0).Max(1).Value("0.3").Step(0.01).Tooltip(
new SliderTooltipData { IsVisible = true, ShowOn = TooltipShowOn.Always,
Format = "P0" , Placement=TooltipPlacement.Before}).Ticks(new SliderTicksData
{ Placement = Placement.After, LargeStep = 0.2, SmallStep = 0.1,
ShowSmallTicks = true, Format = "P0" }).Render()
```

### FORMAT-API.CS

```
public ActionResult FormatAPI()
{
 return View();
}
```



### Using Events

In this method, we will be retrieving the values from the slider events then process them to desired formatted the values. In this sample we have customized the ticks values into weekdays as one formatting and tooltip values into day of the week as another formatting.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(0).Max(6).Value("0").Step(1).Tooltip(new
SliderTooltipData { IsVisible = true, Placement = TooltipPlacement.Before
}).Ticks(new SliderTicksData { Placement = Placement.After, LargeStep = 1
}).TooltipChange("tooltipChangeHandler").RenderingTicks("renderingTicksHandl
er").Render()
<script>
function renderingTicksHandler(args) {
 // Weekdays Array
 var daysArr =
 ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];
 // Customizing each ticks text into weekdays
 args.text = daysArr[parseFloat(args.value)];
}
function tooltipChangeHandler(args) {
 // Customizing tooltip to display the Day (in numeric) of the week
 args.text = 'Day ' + (Number(args.value) + 1).toString();
}
</script>
```

#### EVENTS.CS

```
public ActionResult Events()
{
 return View();
}
```



### Movement Limits and Drag Interval

The slider limits restrict the slider thumb between a particular range. This is used if higher or lower value affects the process or product where the slider is being used.

The following are the six options in the slider's limits object. Each API in the limits object is optional.

- enabled: Enables the limits in the Slider.
- minStart: Sets the minimum limit for the first handle.
- minEnd: Sets the maximum limit for the first handle.
- maxStart: Sets the minimum limit for the second handle.
- maxEnd: Sets the maximum limit for the second handle.

- startHandleFixed: Locks the first handle.
- endHandleFixed: Locks the second handle.

### Default and MinRange Slider limits

There is only one handle in the Default and MinRange Slider, so minStart, minEnd, and startHandleFixed options can be used. When the limits are enabled in the Slider, the limited area becomes darkened. So you can differentiate the allowed and restricted area. Refer to the following snippet to enable the limits in the Slider.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(0).Max(100).Value(25).Type(SliderType.Default).Enabled(true).Tooltip(new SliderTooltipData { IsVisible = true })
.Limits(new SliderLimitData { Enabled = true, MinStart = 10, MinEnd = 40 }).Render()
```

#### DEFAULT-LIMITS.CS

```
public IActionResult Index()
{
 return View();
}
```



### Range Slider limits

In the range slider, both handles can be restricted and locked from the limit's object. In this sample, the first handle is limited between 10 and 40, and the second handle is limited between 60 and 90.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(0).Max(100).Value(ViewBag.range).Type(SliderType.Range).Enabled(true).Tooltip(new SliderTooltipData { IsVisible = true })
.Limits(new SliderLimitData { Enabled = true, MinStart = 10, MinEnd = 40, MaxStart = 60, MaxEnd = 90 }).Render()
```

#### RANGE-LIMITS.CS

```
public IActionResult Index()
{
 ViewBag.range = new int[] { 25, 75 };
}
```

```
 return View();
 }
```



### Handle lock

The movement of slider handles can be locked by enabling the `startHandleFixed` and `endHandleFixed` properties in the limit's object.

In this sample, the movement of both slider handles has been locked.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(0).Max(100).Value(ViewBag.range).Type(SliderType.Range).Enabled(true).Tooltip(new SliderTooltipData { IsVisible = true }).Limits(new SliderLimitData { Enabled = true, StartHandleFixed = true, EndHandleFixed = true, MinStart = 10, MaxEnd = 90 }).Render()
```

### HANDLE-LOCK.CS

```
public IActionResult Index()
{
 ViewBag.range = new int[] { 25, 75 };
 return View();
}
```



### CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the slider track

Use the following CSS to customize the slider track.

```
`css
```

```
.e-control-wrapper.e-slider-container.e-horizontal .e-slider-track {
```

```
background: #007bff;
height: 3px;
}
,
```

#### Customizing the slider handle

Use the following CSS to customize the slider handle properties.

```
`css
.e-control-wrapper.e-slider-container .e-slider .e-handle {
background-color: #f9920b;
border-radius: 50%;
border: 0;
}
,
```

#### Customizing the slider limits

Use the following CSS to customize the slider limits.

```
`css
.e-control-wrapper.e-slider-container.e-horizontal .e-limits {
background-color: rgba(69, 100, 233, 0.46);
}
,
```

#### Customizing the slider ticks

Use the following CSS to customize the slider ticks.

```
`css
.e-scale .e-tick.e-custom::before {
content: '\e967';
position: absolute;
}
,
```

#### Customizing the slider buttons

Use the following CSS to customize the slider buttons.

```
`css
.e-control-wrapper.e-slider-container .e-slider-button {
background: #007bff;
height: 25px;
```

```
width: 25px;
```

```
}
```

```
,
```

### Accessibility in ASP.NET MVC Range Slider component

The Range Slider component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Range Slider component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

```
<style>
```

```
.post .post-content img {
```

```
display: inline-block;
```

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```



<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The Range Slider component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Range Slider component:

| Attributes                    | Purpose                                                                                                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| ---                           | ---                                                                                                                                      |
| <code>role=slider</code>      | Used to convey a significant and contextual message to the user.                                                                         |
| <code>aria-valuemin</code>    | Indicates the Minimum value of the slider.                                                                                               |
| <code>aria-valuemax</code>    | Indicates the Maximum value of the slider.                                                                                               |
| <code>aria-valuenow</code>    | Indicates the current value of the slider.                                                                                               |
| <code>aria-valuetext</code>   | Returns the current text of the slider.                                                                                                  |
| <code>aria-orientation</code> | Indicates whether the Slider is oriented horizontally or vertically.                                                                     |
| <code>aria-label</code>       | Provides an accessible name for the Slider, serving as label text for the Slider's left and right buttons (for increment and decrement). |

### Keyboard interaction

The Range Slider component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Range Slider component.

| Press                 | To do this                                                                                                                                   |
|-----------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| ---                   | ---                                                                                                                                          |
| Right Arrow/Up Arrow  | Increase the Slider value.                                                                                                                   |
| Left Arrow/Down Arrow | Decrease the Slider value.                                                                                                                   |
| Home                  | Moves to the start value (for Range Slider when the second thumb is focused and the Home key is pressed, it moves to the first thumb value). |
| <kbd>End              | Moves to the end value (for Range Slider when the first thumb is focused and the End key is pressed, it moves to the second thumb value).    |
| Page Up               | Increases the Slider by <code>largeStep</code> value.                                                                                        |
| Page Down             | Decreases the Slider by <code>largeStep</code> value.                                                                                        |

### Ensuring accessibility

The Range Slider component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Range Slider component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Range Slider component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## Migration from Essential JS 1

This article describes the API migration process of Slider component from Essential JS 1 to Essential JS 2.

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Max value | **Property:** *MaxValue* <br/><br/> `Html.EJ().Slider("slider").MaxValue(50)` | **Property:** *Max* <br/><br/> `@Html.EJS().Slider("slider").Max(50)` |

| Min value | **Property:** *MinValue* <br/><br/> `Html.EJ().Slider("slider").MinValue(50)` | **Property:** *Min* <br/><br/> `@Html.EJS().Slider("slider").Min(50)` |

| Step | **Property:** *IncrementStep, LargeStep, SmallStep, ShowSmallTicks* <br/><br/> `Html.EJ().Slider("slider").IncrementStep(20).LargeStep(20)<br/>.SmallStep(10).ShowSmallTicks(true).ShowScale(true)` | **Property:** *Ticks* <br/><br/> `@Html.EJS().Slider("slider").Ticks(new SliderTicksData { ShowSmallTicks = true, LargeStep = 20, SmallStep = 5, Placement = Placement.After })` |

| Type | **Property:** *SliderType* <br/><br/> `Html.EJ().Slider("slider").SliderType(SlideType.Range)` | **Property:** *Type* <br/><br/> `@Html.EJS().Slider("slider").Type(SliderType.Range)` |

| Tooltip | **Property:** *ShowTooltip* <br/><br/> `Html.EJ().Slider("slider").ShowTooltip(true)` | **Property:** *Tooltip* <br/><br/> `@Html.EJS().Slider("slider").Tooltip(new SliderTooltipData { Placement = TooltipPlacement.After, IsVisible = true })` |

| RTL | **Property:** *EnableRTL* <br/><br/> `Html.EJ().Slider("slider").EnableRTL(true)` | **Property:** *EnableRtl* <br/><br/> `@Html.EJS().Slider("slider").EnableRtl(true)` |

| Custom values | **Not Applicable** | **Property:** *CustomValues* <br/><br/> `String[] values = { "Mon", "Tue", "Wed" };<br/> ViewBag.customValues = values;<br/> <br/> @Html.EJS().Slider("slider").CustomValues(ViewBag.customValue)` |

| Limit the slider movement | **Not Applicable** | **Property:** *Limits* <br/><br/> `@Html.EJS().Slider("slider").Limits(new SliderLimitData { Enabled = true, MinStart = 20, MinEnd = 40 })` |

| Disable | **Method:** *disable* <br/><br/> `@Html.EJ().Slider("slider") <br/><br/> var sliderObj = $("#slider").ejSlider("instance");<br/> sliderObj.disable();` | **Property:** *Enabled* <br/><br/> `@Html.EJS().Slider("slider") <br/> <br/> var sliderObj = document.getElementById('slider').ej2_instances[0];<br/> sliderObj.enabled = false;` |

| Enable | **Method:** *enable* <br/><br/> `@Html.EJ().Slider("slider") <br/><br/> var sliderObj = $("#slider").ejSlider("instance");<br/> sliderObj.enable();` | **Property:** *Enabled* <br/><br/> `@Html.EJ().Slider("slider").Enabled(false) <br/> <br/> var sliderObj = document.getElementById('slider').ej2_instances[0];<br/> sliderObj.enabled = true;` |

| Set Value | **Method:** `setValue(value,[enableAnimation])` <br/><br/>  
 @Html.EJ().Slider("slider")<br/><br/> var sliderObj = \$("#slider").ejSlider("instance"); <br/>  
 sliderObj.setValue(50); | **Property:** `value` <br/><br/> @Html.EJS().Slider("slider").Enabled(false)  
 <br/> <br/> var sliderObj = document.getElementById('slider').ej2\_instances[0]; <br/>  
 sliderObj.value = 30; |

| Get Value | **Method:** `getValue()` <br/><br/> @Html.EJ().Slider("slider")<br/><br/> var sliderObj =  
 \$("#slider").ejSlider("instance"); <br/> sliderObj.getValue(); | **Property:** `value` <br/><br/>  
 @Html.EJS().Slider("slider").Value(50) <br/><br/> var sliderObj =  
 document.getElementById('slider').ej2\_instances[0]; <br/> var value = sliderObj.value; |

| Destroy | **Not Applicable** | **Method:** `destroy()` <br/><br/>  
 @Html.EJS().Slider("slider").Value(50)<br/><br/> var sliderObj =  
 document.getElementById('slider').ej2\_instances[0]; <br/> sliderObj.destroy(); |

| Change | **Event:** `Change` <br/><br/> @Html.EJ().Slider("slider").ClientSideEvents(e =>  
 e.Change("onChange")) <br/><br/> function onChange(args) { } | **Event:** `Changed` <br/><br/>  
 @Html.EJS().Slider("slider").Changed("onChanged") <br/><br/> function onChanged(args) { } |

| Create | **Event:** `Create` <br/><br/> @Html.EJ().Slider("slider").ClientSideEvents(e =>  
 e.Create("onCreate")) <br/><br/> function onCreate(args) { } | **Event:** `Created` <br/><br/>  
 @Html.EJS().Slider("slider").Created("onCreated") <br/><br/> function onCreated(args) { } |

| Slide | **Event:** `Slide` <br/><br/> @Html.EJ().Slider("slider").ClientSideEvents(e =>  
 e.Slide("onSlide")) <br/><br/> function onSlide(args) { } | **Event:** `Change` <br/><br/>  
 @Html.EJS().Slider("slider").Change("onChange") <br/><br/> function onChange(args) { } |

| Start | **Event:** `Start` <br/><br/> @Html.EJ().Slider("slider").ClientSideEvents(e =>  
 e.Start("onStart")) <br/><br/> function onStart(args) { } | **Event:** `Created` <br/><br/>  
 @Html.EJS().Slider("slider").Created("onCreated") <br/><br/> function onCreated(args) { } |

| Stop | **Event:** `Stop` <br/><br/> @Html.EJ().Slider("slider").ClientSideEvents(e =>  
 e.Stop("onStop")) <br/><br/> function onStop(args) { } | **Event:** `Changed` <br/><br/>  
 @Html.EJS().Slider("slider").Changed("onCreated") <br/><br/> function onChanged(args) { } |

| Rendered Ticks | **Not Applicable** | **Event:** `RenderedTicks` <br/><br/>  
 @Html.EJS().Slider("slider").RenderedTicks("onRenderedTicks") <br/><br/> function  
 onRenderedTicks(args) { } |

## How To

### Time Range Slider

The time formatting can be achieved same as the date formatting using `renderingTicks` and `change` events. The process of time formatting is explained in the below sample.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(1373693400000).Max(1373715000000).Step(360
0000).RenderingTicks("renderingTicksHandler").TooltipChange("tooltipChangeHa
ndler").Tooltip(new SliderTooltipData { Placement = TooltipPlacement.Before,
```

```

isVisible = true }).Ticks(new SliderTicksData { Placement = Placement.After,
LargeStep = 7200000 }).ShowButtons(true).Render()
<script type="text/javascript">
 function tooltipChangeHandler(args) {
 var totalMiliSeconds = Number(args.text);
 var custom = { hour: '2-digit', minute: '2-digit' };
 args.text = new Date(totalMiliSeconds).toLocaleTimeString("en-us",
custom);
 }
 function renderingTicksHandler(args) {
 var totalMiliSeconds = Number(args.value);
 var custom = { hour: '2-digit', minute: '2-digit' };
 args.text = new Date(totalMiliSeconds).toLocaleTimeString("en-us",
custom);
 }
</script>

```

### TIME-FORMAT.CS

```

public ActionResult TimeFormat()
{
 return View();
}

```



### Date Range Slider

The Date formatting can be achieved in ticks and tooltip using `renderingTicks` and `tooltipChange` events respectively. The process of formatting is explained in the below sample.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

@Html.EJS().Slider("default").Min(1371081600000).Max(1371772800000).Step(864
00000).TooltipChange("tooltipChangeHandler").RenderingTicks("renderingTicksH
andler").ShowButtons(true).Tooltip(new SliderTooltipData { Placement =
TooltipPlacement.Before, isVisible = true }).Ticks(new SliderTicksData {
Placement = Placement.After, LargeStep = 172800000, }).Render()
<script>
 function tooltipChangeHandler(args) {
 var totalMiliSeconds = Number(args.text);
 // Converting the current milliseconds to the respective date in
desired format
 var custom = { year: "numeric", month: "short", day: "numeric" };
 args.text = new Date(totalMiliSeconds).toLocaleDateString("en-us",
custom);
 }

```

```

 }
 function renderingTicksHandler(args) {
 var totalMiliSeconds = Number(args.value);
 // Converting the current milliseconds to the respective date in
 desired format
 var custom = { year: "numeric", month: "short", day: "numeric" };
 args.text = new Date(totalMiliSeconds).toLocaleDateString("en-us",
 custom);
 }
</script>

```

### DATE-FORMAT.CS

```

public ActionResult DateFormat()
{
 return View();
}

```



### Numeric Range Slider

The numeric values can be formatted into different decimal digits or fixed number of whole numbers or to represent the units. The Numeric processing is demonstrated below.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div id='kilometer'>

@Html.EJS().Slider("kilometer").Min(0).Max(100).Step(1).Value("30").Tooltip(
new SliderTooltipData { IsVisible = true, Format = "##.## Km" }).Ticks(new
SliderTicksData { Placement = Placement.After, LargeStep = 20, SmallStep =
10, ShowSmallTicks = true, Format = "##.## Km" }).Render()
</div>
<div id='decimal'>

@Html.EJS().Slider("decimalobj").Min(0.1).Max(0.2).Value("0.15").Step(0.01).
Tooltip(new SliderTooltipData { IsVisible = true, Format = "##.##00"
}).Ticks(new SliderTicksData { Placement = Placement.After, LargeStep =
0.02, SmallStep = 0.01, ShowSmallTicks = true, Format = "##.##00" }).Render()
</div>
<div id='slider'>

@Html.EJS().Slider("sliderobj").Min(0).Max(100).Value("5").Step(1).Tooltip(n
ew SliderTooltipData { IsVisible = true, Format = "##.## Km" }).Ticks(new
SliderTicksData { Placement = Placement.After, LargeStep = 20, SmallStep =
10, ShowSmallTicks = true, Format = "##.## Km" }).Render()
</div>

```

**NUMERIC-VALUE.CS**

```
public ActionResult NumericValue()
{
 return View();
}
```

Slider formatted with unit representation



Slider formatted with three decimal specifiers



Slider formatted with two leading zeros

**Customize Slider Bar**

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the slider bar. The slider bar can be customized with different themes. By default, slider have class name `e-slider-track` for bar. The class can be overridden with our own color values like the following code snippet.

```
`css
```

```
.e-control.e-slider .e-slider-track .e-range {
```

```
background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%,
#e33df9 83%, #e14423 100%);
```

```
}
```

```
`
```

You can also apply background color for a certain range depending upon slider values, using change event.

```
`javascript
```

```
function change(args) {
```

```
if (args.value > 0 && args.value <= 25) {
```

```
// Change handle and range bar color to green when
```

```

(slidebarHandle).style.backgroundColor = 'green';
(slidebarTrack).style.backgroundColor = 'green';
} else if (args.value > 25 && args.value <= 50) {
// Change handle and range bar color to royal blue
(slidebarHandle).style.backgroundColor = 'royalblue';
(slidebarTrack).style.backgroundColor = 'royalblue';
} else if (args.value > 50 && args.value <= 75) {
// Change handle and range bar color to dark orange
(slidebarHandle).style.backgroundColor = 'darkorange';
(slidebarTrack).style.backgroundColor = 'darkorange';
} else if (args.value > 75 && args.value <= 100) {
// Change handle and range bar color to red
(slidebarHandle).style.backgroundColor = 'red';
(slidebarTrack).style.backgroundColor = 'red';
}
}
,

```

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div id='container'>
 <div class="col-lg-12 control-section">
 <div class="slider-content-wrapper">
 <div class="slider_container">
 <div class="slider-labeltext slider_userselect">Height</div>
 @Html.EJS().Slider("height_slider").Render()
 </div>
 <div class="slider_container">
 <div class="slider-labeltext slider_userselect">Gradient
color</div>
 @Html.EJS().Slider("gradient_slider").Type(SliderType.MinRange).Render()
 </div>
 <div class="slider_container">
 <div class="slider-labeltext slider_userselect">Dynamic
thumb and selection bar color</div>
 @Html.EJS().Slider("dynamic_color_slider").Type(SliderType.MinRange).Change(
"changeHandler").Created("createdHandler").Render()
 </div>
 </div>
 </div>
</div>
<script type="text/javascript">

```

```

var sliderTrack;
var sliderHandle;
function changeHandler(args) {
 if (args.value > 0 && args.value <= 25) {
 // Change handle and range bar color to green when
 (sliderHandle).style.backgroundColor = 'green';
 (sliderTrack).style.backgroundColor = 'green';
 } else if (args.value > 25 && args.value <= 50) {
 // Change handle and range bar color to royal blue
 (sliderHandle).style.backgroundColor = 'royalblue';
 (sliderTrack).style.backgroundColor = 'royalblue';
 } else if (args.value > 50 && args.value <= 75) {
 // Change handle and range bar color to dark orange
 (sliderHandle).style.backgroundColor = 'darkorange';
 (sliderTrack).style.backgroundColor = 'darkorange';
 } else if (args.value > 75 && args.value <= 100) {
 // Change handle and range bar color to red
 (sliderHandle).style.backgroundColor = 'red';
 (sliderTrack).style.backgroundColor = 'red';
 }
}

function createdHandler(args) {
 sliderTrack =
document.getElementById('dynamic_color_slider').querySelector('.e-range');
 sliderHandle =
document.getElementById('dynamic_color_slider').querySelector('.e-handle');
 (sliderHandle).style.backgroundColor = 'green';
 (sliderTrack).style.backgroundColor = 'green';
}
</script>
<style>
.slider-content-wrapper {
 width: 40%;
 margin: 0 auto;
 min-width: 185px;
}
.slider-userselect {
 -webkit-user-select: none;
 /* Safari 3.1+ */
 -moz-user-select: none;
 /* Firefox 2+ */
 -ms-user-select: none;
 /* IE 10+ */
 user-select: none;
 /* Standard syntax */
}
.slider-labeltext {
 text-align: left;
 font-weight: 500;
 font-size: 13px;
 padding-bottom: 10px;
}
#height_slider .e-handle,
#gradient_slider .e-handle {
 height: 20px;
 width: 20px;
 margin-left: -10px;
}

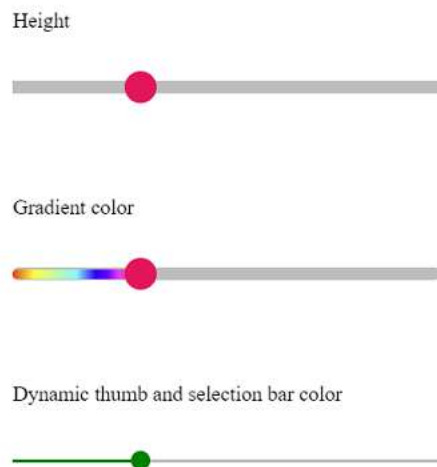
```



```
 top: calc(50% - 10px);
 }
 .slider_container {
 margin-top: 40px;
 }
 #height_slider .e-tab-handle::after {
 background-color: #f9920b;
 }
 #height_slider .e-slider-track {
 height: 8px;
 top: calc(50% - 4px);
 border-radius: 0;
 }
 #gradient_slider .e-range {
 height: 6px;
 top: calc(50% - 3px);
 border-radius: 5px;
 background: -webkit-gradient(linear, left top, left bottom, color-stop(0%, #1e5799), color-stop(100%, #7db9e8));
 background: -webkit-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
 background: linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
 background: -moz-linear-gradient(left, #e1451d 0, #fdff47 17%, #86f9fe 50%, #2900f8 65%, #6e00f8 74%, #e33df9 83%, #e14423 100%);
 }
 #gradient_slider .e-slider-track {
 height: 8px;
 top: calc(50% - 4px);
 border-radius: 5px;
 }
</style>
```

### CUSTOM-BAR.CS

```
public ActionResult TimeFormat()
{
 return View();
}
```



### Customize Slider Limits

Slider appearance can be customized via CSS. By overriding the slider CSS classes, the slider limit bar can be customized.

Here, the limit bar is customized with different background color. By default, the slider has class `e-limits` for limits bar.

You can override the class with our own color values as given in the following code snippet.

```
`css
.e-slider-container.e-horizontal .e-limits {
background-color: rgba(69, 100, 233, 0.46);
}
```

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div id='container'>
 <div class="col-lg-8 control-section">
 <div class="content-wrapper">
 <div class="sliderwrap">
 <label class="userselect">MinRange Slider With
Limits</label>

@Html.EJS().Slider("minrange").Min(0).Max(100).Value(25).Type(SliderType.Default).Enabled(true).Ticks(new SliderTicksData { Placement =
Placement.Before, LargeStep = 20, SmallStep = 5, ShowSmallTicks = true
}).Tooltip(new SliderTooltipData { IsVisible = true, Placement =
TooltipPlacement.Before, ShowOn = TooltipShowOn.Focus }).Limits(new
SliderLimitData { Enabled = true, MinStart = 10, MinEnd = 40 }).Render()
 </div>
 <div class="sliderwrap">
 <label class="userselect">Range Slider With
Limits</label>
```

```

@Html.EJS().Slider("range").Min(0).Max(100).Value(ViewBag.rangeValue).Type(S
liderType.Range).Enabled(true).Ticks(new SliderTicksData { Placement =
Placement.Before, LargeStep = 20, SmallStep = 5, ShowSmallTicks = true
}).Tooltip(new SliderTooltipData { IsVisible = true, Placement =
TooltipPlacement.Before, ShowOn = TooltipShowOn.Focus }).Limits(new
SliderLimitData { Enabled = true, MinStart = 10, MinEnd = 40, MaxStart = 60,
MaxEnd = 90}).Render()
</div>
</div>
</div>
</div>
<style>
.content-wrapper {
 width: 52%;
 margin: 0 auto;
 min-width: 185px;
}
.sliderwrap {
 margin-top: 45px;
}
.e-bigger .content-wrapper {
 width: 80%;
}
.sliderwrap label {
 padding-bottom: 50px;
 font-size: 13px;
 font-weight: 500;
 margin-top: 15px;
 display: block;
}
.userselect {
 -webkit-user-select: none;
 /* Safari 3.1+ */
 -moz-user-select: none;
 /* Firefox 2+ */
 -ms-user-select: none;
 /* IE 10+ */
 user-select: none;
 /* Standard syntax */
}
.property-custom td {
 padding: 5px;
}
#range .e-limits,
#minrange .e-limits {
 background-color: #ccc;
 background-color: rgba(69, 100, 233, 0.46);
}
</style>

```

### CUSTOM-LIMITS.CS

```

public ActionResult TimeFormat()
{
 ViewBag.rangeValue = new int[] { 25, 75 };
}

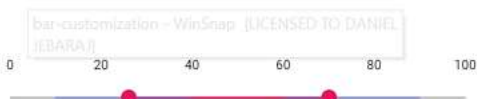
```

```
return View();
}
```

MinRange Slider With Limits



Range Slider With Limits



### Customize Slider Ticks label

Slider view can be customized via CSS. By overriding the slider CSS classes, you can customize the ticks. The ticks in slider allows you to easily identify the current value/values of the slider. It contains [smallStep](#) and [largeStep](#). By default, slider has class `e-tick` for slider ticks. You can override the class as per your requirement. Refer to the following code snippet to render ticks.

```
`css
.e-scale .e-tick.e-custom::before {
content: '\e967';
position: absolute;
}
`
```

Here, the color for rendered ticks has been applied through `nth-child(childnumber)`. *The color is applied to the value of the childnumber in the slider.*

```
`css
ticks_slider .e-scale :nth-child(1)::before {
color: red;
}
`
```

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div id='container'>
 <div class="col-lg-12 control-section">
 <div class="slider-content-wrapper">
 <div class="slider_container" id="slider_wrapper">
 <div class="slider_labelText userselect">Dynamic ticks
color</div>
 <!-- Ticks slider element -->
 </div>
 </div>
 </div>
</div>
```

```

@Html.EJS().Slider("ticks_slider").Type(SliderType.MinRange).Value(30).Enabled(true).Step(5).Ticks(new SliderTicksData { Placement = Placement.Before, LargeStep = 20 }).RenderingTicks("renderingTicksHandler_1").Render()
 </div>
 <div class="slider_container">
 <div class="slider_labelText userselect">Ticks with
legends</div>
 <!-- Ticks slider element -->

@Html.EJS().Slider("slider").Value(30).Type(SliderType.MinRange).Enabled(true).Step(5).Ticks(new SliderTicksData { Placement = Placement.Both, LargeStep = 20, SmallStep=5 }).RenderingTicks("renderingTicksHandler_2").Render()
 </div>
</div>
</div>
</div>
</div>
<script>
function renderingTicksHandler_1(args) {
 if (args.tickElement.classList.contains('e-large')) {
 args.tickElement.classList.add('e-custom');
 }
}
function renderingTicksHandler_2(args) {
 var li = args.ticksWrapper.getElementsByClassName('e-large');
 var remarks = ['Very Poor', 'Poor', 'Average', 'Good', 'Very Good', 'Excellent'];
 for (var i = 0; i < li.length; ++i) {
 (li[i].querySelectorAll('.e-tick-both')[1]).innerText = remarks[i];
 }
}
</script>
<style>
.slider-content-wrapper {
 width: 40%;
 margin: 0 auto;
 min-width: 185px;
}
.userselect {
 -webkit-user-select: none;
 /* Safari 3.1+ */
 -moz-user-select: none;
 /* Firefox 2+ */
 -ms-user-select: none;
 /* IE 10+ */
 user-select: none;
 /* Standard syntax */
}
.slider_labelText {
 text-align: left;
 font-weight: 500;
 font-size: 13px;
 padding-bottom: 40px;
}
.slider_container {
 margin-top: 40px;
}

```

```

}
.e-bigger .slider-content-wrapper {
 width: 80%;
}
#ticks_slider .e-range {
 z-index: unset;
}
/* csslint ignore:start */
#ticks_slider .e-scale .e-tick {
 background-image: none;
 visibility: visible;
 font-family: 'e-customized-icons';
}
@@font-face {
 font-family: 'e-customized-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj8iS4cAAAEoAAAAVmNtYXDS5tJrAAABjAAAAEBnbHlmdMA
KbQAAAdQAAAOwAGVhZBNseyYAAADQAAAAANmhoZWEHogNjAAAArAAAACRobXR4C9AAAAAAYAAAAA
MbG9jYQCaAdgAAAHMAAAACG1heHABEAeEuAAABCAAAACBuYW1lc0cOBgAABYQAAAIlcG9zdNSlKbQ
AAAEsAAAAARwABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAtxzLE18
PPPUACwPoAAAAANgtmycAAAAA2C2bJwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAADASIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQpWAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA6QLpZwNS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAAAAAAAgAAAAAUAAMAAQAAABQABAAsAAAAABgAEAAEAukC6Wf//wAA6QLpZ//AAAAAA
BAAYABgAAAAEAAgAAAAAamgHYAAIAAAAAA+oD6gAzAICAAAEzHxghNT8WEx8THQEPEisBLxI9AT8
SAGAQECQmKCgPKScTEhIREA8ODQwKCgQHBQQBAfwqAQMFbgcKCgWNDg8QERISEycpKSgoJiQgDQw
MDAwXFhUUEhEPDQsJCAIDAQEBAQMCCakLDQ8REhQVFhCMDAwMDQ0MDAwMFxYVFBIRDw0LCQgCAWE
BAQEDAggJCw0PERIUFRYXDAwMDAGFAQMEBwkKDQ4ICakKCgoLCwwMDAcNDg8Og3sPDw4NDgwMDAs
LCgoKCQgIDg0KCQCeAwJnAQEBAGMHCgsNDxESExUWFfwMDQwNDA0MDAwXFhUTExAPDQwJBwMCAgE
BAgIDBwkMDQ8QExMVfHcMDAwNDA0MDQwMFxYVExIRDw0LCgcDAgEBAAAAAwAAAAAD8wPzAF8AwAE
hAAABDxmFfZ8XLxcPAjcfFA8XLxc/Fx8CJw8UHxc/Fy8XDwIBqRQUFBISERAQDg0NCwoJBwcFBAI
BAQIEBQcHCQoLDQ0OEBAREhIUFBQVFhYWFhYWFRUTFBISERAQDg0NCwoJBwcFBAIQAIEBQcHCQo
LDQ0OEBAREhIUExUVFhYWFhYWTg4NGxkZGBYWFRMSEA8OCwsIBwUDAQEDBQcICwsODxASExUWFhg
ZGRsbHB0dHh4dHRwbGxkZGBYWFRMSEA8NDAsIBwUDAQEDBQcICwsODxASExUVFhgZGRsbHB0dHh4
dHd0QDx4eHBsaGRcWFRIREA0MCQgGAWEBawYICQwNEBESFRYXGRobHB4eHyEgIiIiIiAhHx4eHBs
aGRcWFRIREA0MCQgGAWEBawYICQwNEBESFRYXGRobHB4eHyEgIiIiIiEDPAYICQoLDQ0OEBAREhI
TFBUVFRYXfYhYWFRQUFBISERAQDg0MDAoJBwcFBAIQAIEBQcHCQoMDA0OEBAREhIUFBQVFhYWFxY
VFRUUEXISERAQDg0NCwoJCAYFBAIQAIEZAQECgWODxASExUVFhgYGHsbHB0dHh4dHRwbGxkZGBY
WFBQSEA8NDAsIBwUDAQEDBQcICwsODxASExUWFhgZGRsbHB0dHh4dHRwbGxoYGBcVFRMSEA8OCws
IBwUDAQEDBTYFBQwNEBESFRYXGRobHB0fHyEgIiIiIiEgHx4eHBsaGRcWFBMRDw4MCQgGAWEBawY
ICQWODxETFBYXGRobHB4eHyEgIiIiIiAhHx4eHBsaGRcWFRIRDw4MCQgGAWEBawYAAAAAAAAASAN4
AAQAAAAAAAAABAAAAQAAAAAAAAQAHAAEAQAAAAAAAAAgAHAAGAAQAAAAAAAAwAHAA8AAQAAAAAAAABAA
HABYAAQAAAAAAAAABQALAB0AAQAAAAAAAAABgAHACgAAQAAAAAACgAsAC8AAQAAAAAACwASAFsAAwABBak
AAAACAG0AAwABBakAAQAOAG8AAwABBakAAgAOAH0AAwABBakAAwAOAIsAAwABBakABAAOAJkaAwA
BBakABQAWAKcAAwABBakABgAOAL0AAwABBakACgBYAMsAAwABBakACwAkASMgZS1pY29uc1JlZ3V
sYXJlLW1jb25zZS1pY29uc1ZlcnNpb24gMS4wZS1pY29uc0ZvbnQgZ2ZuZXJhdGVkIHVzaW5nIFN
5bmNmdXNpb24gTWV0cm8gU3RlZGlvd3d3LnN5bmNmdXNpb24uY29tACAAZQAtAGkAYwBvAG4AcwB
SAGUaZwB1AGwAYQByAGUALQBpAGMabwBuAHMAZQAtAGkAYwBvAG4AcwBWAGUAcgBzAGkAbwBuACA
AMQAUaDAAZQAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQByAGEaDABLAGQAIAAB1AHMAAQb
uAGCAIAABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAQcBvACAAUwB0AHUAZABpAG8AdwB3AHc
ALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAAAAKAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAADAQIBAwEEAAh0ZW1wLWN1cxJGQl9DaGVja2JveF9zZWx1Y3QAAAA=)
format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:end */

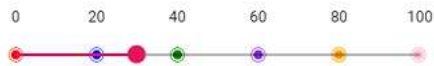
```

```
#ticks_slider .e-scale {
 z-index: 0;
}
#ticks_slider .e-scale .e-custom::before {
 content: '\e967';
 position: absolute;
}
#ticks_slider .e-scale :nth-child(1)::before {
 color: red;
}
#ticks_slider .e-scale :nth-child(2)::before {
 color: blue;
}
#ticks_slider .e-scale :nth-child(3)::before {
 color: green;
}
#ticks_slider .e-scale :nth-child(4)::before {
 color: blueviolet;
}
#ticks_slider .e-scale :nth-child(5)::before {
 color: orange;
}
#ticks_slider .e-scale :nth-child(6)::before {
 color: pink;
}
#ticks_slider .e-scale .e-custom::before {
 font-size: 10px;
}
#ticks_slider .e-scale .e-custom::before {
 top: calc(50% + 1px);
 left: calc(50% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(1)::before {
 top: calc(50% + 1px);
 left: calc(0% - 5px);
}
#slider_wrapper #ticks_slider .e-scale :nth-child(6)::before {
 top: calc(50% + 1px);
 left: calc(100% - 6px);
}
</style>
```

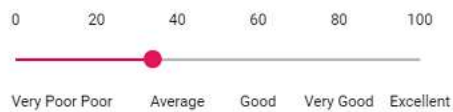
### CUSTOM-TICKS.CS

```
public ActionResult TimeFormat()
{
 return View();
}
```

Dynamic ticks color



Ticks with legends



### Customize Slider Thumb

Slider appearance can be customized through CSS. By overriding the slider CSS classes, you can customize the thumb. By default, slider has unique class `e-handle` for slider thumb. You can override the following class as per your requirement. Here, in the sample, the slider thumb has been customized to square, circle, oval shapes, and background image has also been customized.

```
`css
```

```
.e-control.e-slider .e-handle {
background-image: url('https://ej2.syncfusion.com/demos/src/slider/images/thumb.png');
background-color: transparent;
height: 25px;
width: 25px;
}

square_slider.e-control.e-slider .e-handle {
border-radius: 0%;
background-color: #f9920b;
border: 0;
}

circle_slider.e-control.e-slider .e-handle {
border-radius: 50%;
background-color: #f9920b;
border: 0;
}

oval_slider.e-control.e-slider .e-handle {
height: 25px;
width: 8px;
top: 3px;
```



```
border-radius: 15px;
background-color: #f9920b;
}
,
```

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div id='container'>
 <div class="col-lg-12 control-section">
 <div class="slider-content-wrapper">
 <div class="slider_container">
 <div class="labelText slider-userselect">Square</div>
 <!-- Square slider element -->

@Html.EJS().Slider("square_slider").Min(0).Max(100).Value(30).Enabled(true).
Render()

 </div>
 <div class="slider_container">
 <div class="labelText slider-userselect">Circle</div>
 <!-- Circle slider element -->

@Html.EJS().Slider("circle_slider").Min(0).Max(100).Value(30).Enabled(true).
Render()

 </div>
 <div class="slider_container">
 <div class="labelText slider-userselect">Oval</div>
 <!-- Oval slider element -->

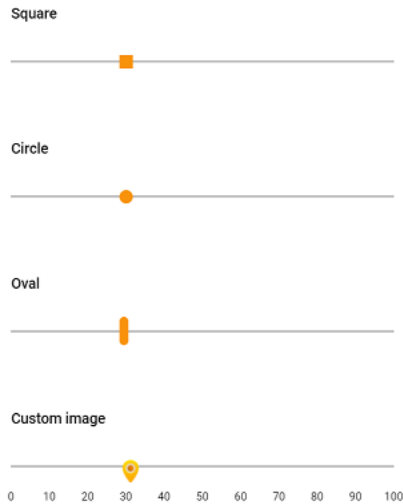
@Html.EJS().Slider("oval_slider").Min(0).Max(100).Value(30).Enabled(true).Re
nder()

 </div>
 <div class="slider_container">
 <div class="labelText slider-userselect">Custom
image</div>
 <!-- Image slider element -->
 @Html.EJS().Slider("image_slider").Value(30).Tooltip(new
SliderTooltipData { IsVisible = true, Placement =
TooltipPlacement.After}).Render()
 </div>
 </div>
 </div>
</div>
<style>
.slider-content-wrapper {
 width: 40%;
 margin: 0 auto;
 min-width: 185px;
}
.slider-userselect {
 -webkit-user-select: none;
 /* Safari 3.1+ */
 -moz-user-select: none;
 /* Firefox 2+ */
```

```
-ms-user-select: none;
/* IE 10+ */
user-select: none;
/* Standard syntax */
}
.labelText {
 text-align: left;
 font-weight: 500;
 font-size: 13px;
 padding-bottom: 10px;
}
.slider_container {
 margin-top: 40px;
}
.e-bigger .content-wrapper {
 width: 80%;
}
#square_slider .e-handle {
 border-radius: 0;
 background-color: #f9920b;
 border: 0;
}
#circle_slider .e-handle {
 background-color: #f9920b;
 border-radius: 50%;
 border: 0;
}
#image_slider .e-handle {
 height: 25px;
 width: 24px;
 background-size: 24px;
}
#image_slider .e-handle {
 background-image:
url('https://ej2.syncfusion.com/demos/src/slider/images/thumb.png');
 background-repeat: no-repeat;
 background-color: transparent;
 border: 0;
}
#square_slider .e-tab-handle::after,
#circle_slider .e-tab-handle::after {
 background-color: #f9920b;
}
#image_slider .e-tab-handle::after {
 background-color: transparent;
}
#oval_slider .e-handle {
 height: 25px;
 width: 8px;
 top: 3px;
 border-radius: 15px;
 background-color: #f9920b;
}
</style>
```

### CUSTOM-THUMB.CS

```
public ActionResult TimeFormat()
{
 return View();
}
```



### Form Slider with FormValidator

The Slider control can be validated using our [FormValidator](#). The following steps walk-through slider validation.

- Render slider control inside a form.
- Bind [changed](#) event in the slider control to validate the slider value when the value changes.
- Initialize and render FormValidator for the form using form ID.
- Set the required property in the FormValidator [rules](#) collection. Here, the [min](#) property of slider that sets the minimum value in the slider control is set, and it has hidden input as enable [validateHidden](#) property is set to true.

**Note:** Form validation is done either by ID or name value of the slider control. Above ID of the slider is used to validate it.

Using slider name: Render slider with name attribute. In the following code snippet, name attribute value of slider is used for form validation.

- Validate the form using [validate](#) method, and it validates the slider value with the defined rules collection and returns the result. If user selects the value less than the minimum value, form will not submit.
- Slider validation can be done during value changes in slider. Refer to the following code snippet.

```
`javascript
// change event handler for slider
function onChanged(args) {
 formObj.validate();
}
```

```
}
,
```

The **FormValidator** has following default validation rules, which are used to validate the Slider control.

| Rules | Description                                                                                                 | Example |
|-------|-------------------------------------------------------------------------------------------------------------|---------|
| max   | Slider control must have value less than or equal to max number   if max: 3, 3 is valid and 4 is invalid    |         |
| min   | Slider control must have value greater than or equal to min number   if min: 4, 5 is valid and 2 is invalid |         |
| regex | Slider control must have valid value in regex format   if regex: '/4/', 4 is valid and 1 is invalid         |         |
| range | Slider control must have value between range number   if range: [4,5], 4 is valid and 6 is invalid          |         |

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
<div id='container'>
 <div class="col-lg-12 control-section">
 <div class="content-wrapper" style="margin-bottom: 25px;overflow-x: hidden">
 <div class="form-title">
 Min
 </div>
 <form id="formMinId" class="form-horizontal">
 <div class="form-group">
 <div class="e-float-input">
 @Html.EJS().Slider("min-
slider").Value(30).Enabled(true).Changed("onMinChanged").Ticks(new
SliderTicksData { Placement = Placement.Before, LargeStep = 20, SmallStep =
5, ShowSmallTicks = true }).Render()
 </div>
 </div>
 </form>
 <div class="form-title">
 Max
 </div>
 <form id="formMaxId" class="form-horizontal">
 <div class="form-group">
 <div class="e-float-input">
 @Html.EJS().Slider("max-
slider").Value(30).Enabled(true).Changed("onMaxChanged").Ticks(new
SliderTicksData { Placement = Placement.Before, LargeStep = 20, SmallStep =
5, ShowSmallTicks = true }).Render()
 </div>
 </div>
 </form>
 <div class="form-title">
 Value
 </div>
```

```

 <form id="formValId" class="form-horizontal">
 <div class="form-group">
 <div class="e-float-input">
 @Html.EJS().Slider("val-
slider").Value(30).Enabled(true).Changed("onValChanged").Ticks(new
SliderTicksData { Placement = Placement.Before, LargeStep = 20, SmallStep =
5, ShowSmallTicks = true }).Render()
 </div>
 </div>
 </form>
 <div class="form-title">
 Range
 </div>
 <form id="formRangeId" class="form-horizontal">
 <div class="form-group">
 <div class="e-float-input">
 @Html.EJS().Slider("range-
slider").Value(30).Enabled(true).Changed("onRangeChanged").Ticks(new
SliderTicksData { Placement = Placement.Before, LargeStep = 20, SmallStep =
5, ShowSmallTicks = true }).Render()
 </div>
 </div>
 </form>
 <div class="form-title">
 Custom
 </div>
 <form id="formCustomId" class="form-horizontal">
 <div class="form-group">
 <div class="e-float-input">
 @Html.EJS().Slider("custom-
slider").Type(SliderType.Range).Enabled(true).Changed("onCustomChanged").Tic
ks(new SliderTicksData { Placement = Placement.Before, LargeStep = 20,
SmallStep = 5, ShowSmallTicks = true }).Render()
 </div>
 </div>
 </form>
 </div>
</div>
<script type="text/javascript">
 var minOptions = {
 rules: {
 'min-slider': {
 validateHidden: true,
 min: [40, "You must select value greater than or equal to
40"]
 }
 }
 };
 var formMinId = document.getElementById('formMinId');
 // Initialize Form validation
 var formMinObj = new ej.inputs.FormValidator(formMinId, minOptions);
 function onMinChanged(args) {
 // validate the slider value in the form
 formMinObj.validate();
 }
 var maxOptions = {

```

```

 rules: {
 'max-slider': {
 validateHidden: true,
 max: [40, "You must select value less than or equal to 40"]
 }
 }
 };
 var formMaxId = document.getElementById('formMaxId');
 // Initialize Form validation
 var formMaxObj = new ej.inputs.FormValidator(formMaxId, maxOptions);
 function onMaxChanged(args) {
 // validate the slider value in the form
 formMaxObj.validate();
 }
 var valOptions = {
 rules: {
 'val-slider': {
 validateHidden: true,
 regex: [/40/, "You must select value equal to 40"]
 }
 }
 };
 var formValId = document.getElementById('formValId');
 // Initialize Form validation
 var formValObj = new ej.inputs.FormValidator(formValId, valOptions);
 function onValChanged(args) {
 // validate the slider value in the form
 formValObj.validate();
 }
 var rangeOptions = {
 rules: {
 'range-slider': {
 validateHidden: true,
 range: [40, 80, "You must select values between 40 and 80"]
 }
 }
 };
 var formRangeId = document.getElementById('formRangeId');
 // Initialize Form validation
 var formRangeObj = new ej.inputs.FormValidator(formRangeId,
rangeOptions);
 function onRangeChanged(args) {
 // validate the slider value in the form
 formRangeObj.validate();
 }
 var customOptions = {
 rules: {
 'custom-slider': {
 validateHidden: true,
 range: [validateRange, "You must select values between 40
and 80"]
 }
 }
 };
 var formCustomId = document.getElementById('formCustomId');
 // Initialize Form validation

```

```

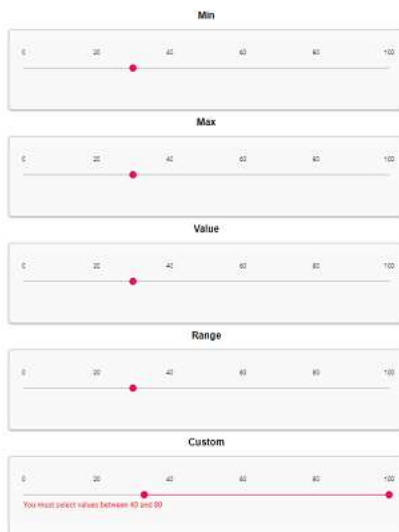
 var formCustomObj = new ej.inputs.FormValidator(formCustomId,
customOptions);
 function onCustomChanged(args) {
 // validate the slider value in the form
 formCustomObj.validate();
 }
 function validateRange(args) {
 var SliderCustomObj = document.getElementById('custom-
slider').ej2_instances[0];
 return SliderCustomObj.value[0] >= 40 && SliderCustomObj.value[1] <=
80;
 }
</script>
<style>
.highcontrast form {
 background: #000000
}
/* csslint ignore:start */
.highcontrast label.e-custom-label,
.highcontrast label.e-float-text,
.highcontrast label.salary,
.highcontrast input::placeholder,
.highcontrast .e-float-input label.e-float-text {
 color: #fff !important;
 line-height: 2.3;
}
/* csslint ignore:end */
.e-error,
.e-float-text {
 font-weight: 500;
}
table,
td,
th {
 padding: 5px;
}
.form-horizontal .form-group {
 margin-left: 0;
 margin-right: 0;
}
form {
 border: 1px solid #ccc;
 box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
 border-radius: 5px;
 background: #f9f9f9;
 padding: 23px;
 padding-bottom: 20px;
 margin: auto;
 max-width: 650px;
}
.form-title {
 width: 100%;
 text-align: center;
 padding: 10px;
 font-size: 16px;
 font-weight: 600;
 color: black;

```

```
}
</style>
```

### FORM-VALIDATOR.CS

```
public ActionResult TimeFormat()
{
 return View();
}
```



### Show Slider from Hidden State

This section demonstrates how-to render the Slider component in hidden state and make it visible in button click. We can initialize Slider in hidden state by setting display as none to it.

In the sample, by clicking on the button, we can make the Slider visible from hidden state, and we must also call the `refresh` method of the Slider to render it properly based on its original dimensions.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs
@Html.EJS().Button("element").Content("Button").Render()
<div id='container' style="margin-top:150px;">

 @Html.EJS().Slider("default").Min(1373693400000).Max(1373715000000).
 Step(3600000).RenderingTicks("renderingTicksHandler").TooltipChange("tooltip
 ChangeHandler").Tooltip(new SliderTooltipData { Placement =
 TooltipPlacement.Before, IsVisible = true }).Ticks(new SliderTicksData {
 Placement = Placement.After, LargeStep = 7200000
 }).ShowButtons(true).Render()
</div>
<script type="text/javascript">
 function tooltipChangeHandler(args) {
 var totalMilliseconds = Number(args.text);
 var custom = { hour: '2-digit', minute: '2-digit' };
 args.text = new Date(totalMilliseconds).toLocaleTimeString("en-us",
 custom);
 }
```



```

 }
 function renderingTicksHandler(args) {
 var totalMilliseconds = Number(args.value);
 var custom = { hour: '2-digit', minute: '2-digit' };
 args.text = new Date(totalMilliseconds).toLocaleTimeString("en-us",
custom);
 }
 document.getElementById('element').onclick = function () {
 //slider visible from hidden state
 slider = document.getElementById("container");
 ticks = document.getElementById("default");
 slider.style.display = "block";
 ticks.ej2_instances[0].refresh();
 };
</script>
<style>
/* render slider in hidden state */
#container{
 display: none;
}
</style>

```

### HIDDEN-SLIDER.CS

```

public ActionResult HiddenSlider()
{
 return View();
}

```



### Reversible Range Slider

You can create a Range Slider rendered with values in reverse order by setting the `min` property to the maximum value and the `max` property to the minimum value. An example of how to achieve a reversible Range Slider is shown below

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Inputs

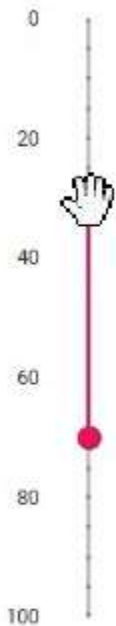
@Html.EJS().Slider("reversible").Min(100).Max(0).Value(ViewBag.range).Enabled(true).Type(SliderType.Range).Orientation(SliderOrientation.Vertical).Ticks(new SliderTicksData { Placement = Placement.Before, LargeStep = 20,

```

```
SmallStep = 5, ShowSmallTicks = true }).Tooltip(new SliderTooltipData {
 IsVisible = true, Placement = TooltipPlacement.Before }).Render()
```

### REVERSIBLE-SLIDER.CS

```
public ActionResult Reverse()
{
 ViewBag.range = new int[] { 30, 70 };
 return View();
}
```



Reversible order can be achieved with **Horizontal** orientation Range Slider by setting **enableRtl** as true.

## Rating

### Getting Started with ASP.NET MVC Rating Control

This section briefly explains about how to include **ASP.NET MVC Rating** control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Rating control

Now, add the Syncfusion ASP.NET MVC Rating control in `~/Home/Index.cshtml` page.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Rating control will be rendered in the default web browser.



### Value

You can set the rating value by using the [Value](#) property.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating1").Value(3).Render()
```



### Precision Modes in ASP.NET MVC Rating Control

You can use the [Precision](#) property of the ASP.NET MVC Rating control to provide ratings with varying levels of precision.

The precision types of Rating are as follows:

- Full: The rating is increased in whole number increments. For example, if the current rating is 2, the next possible ratings are 3, 4, and so on.
- Half: The rating is increased in increments of 0.5 (half). For example, if the current rating is 2.5, the next possible ratings are 3, 3.5, 4, and so on.
- Quarter: The rating is increased in increments of 0.25 (quarter). For example, if the current rating is 3.75, the next possible ratings are 4, 4.25, 4.5, and so on.
- Exact: The rating is increased in increments of 0.1. For example, if the current rating is 3.9, the next possible ratings are 4, 4.1, 4.2, and so on.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
```

```

<label>Full Precision</label>

@Html.EJS().Rating("rating").Value(3).Precision(PrecisionType.Full).Render()

<label>Half Precision</label>

@Html.EJS().Rating("rating1").Value(2.5).Precision(PrecisionType.Half).Render()

<label>Quarter Precision</label>

@Html.EJS().Rating("rating2").Value(3.75).Precision(PrecisionType.Quarter).Render()

<label>Exact Precision</label>

@Html.EJS().Rating("rating3").Value(2.3).Precision(PrecisionType.Exact).Render()

```

### PRECISIONMODES.CS

```

public ActionResult PrecisionModes()
{
 return View();
}

```

Full Precision



Half Precision



Quarter Precision



Exact Precision



### Labels in ASP.NET MVC Rating Control

You can use the [ShowLabel](#) property to display a label that shows the current value of the rating. When the `ShowLabel` property is set to `true`, a label will be displayed.

### CSHTML

```

@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).ShowLabel(true).Render()

```

### SHOWLABEL.CS

```

public ActionResult ShowLabel()
{
 return View();
}

```



### Label position

The Rating control allows you to place the label on the top, bottom, left, or right side of the rating using the [LabelPosition](#) property.

The following label positions are supported:

- Top: The label is placed on the top of the rating.
- Bottom: The label is placed on the bottom of the rating.
- Left: The label is placed on the left side of the rating.
- Right: The label is placed on the right side of the rating.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
<label>Left Label Position</label>

@Html.EJS().Rating("rating1").Value(3).ShowLabel(true).LabelPosition(LabelPosition.Left).Render()

<label>Right Label Position</label>

@Html.EJS().Rating("rating2").Value(3).ShowLabel(true).LabelPosition(LabelPosition.Right).Render()

<label>Top Label Position </label>

@Html.EJS().Rating("rating3").Value(3).ShowLabel(true).LabelPosition(LabelPosition.Top).Render()

<label>Bottom Label Position</label>

@Html.EJS().Rating("rating4").Value(3).ShowLabel(true).LabelPosition(LabelPosition.Bottom).Render()
```

### LABELPOSITION.CS

```
public ActionResult LabelPosition()
{
 return View();
}
```

**Left Label Position**

3 / 5 ★ ★ ★ ☆ ☆

**Right Label Position**

★ ★ ★ ☆ ☆ 3 / 5

**Top Label Position**

3 / 5

★ ★ ★ ☆ ☆

**Bottom Label Position**

★ ★ ★ ☆ ☆

3 / 5

**Label template**

You can use the [LabelTemplate](#) tag directive to specify a custom template for the **Label** of the rating.

The current value of the rating will be passed as the **value** property in the template context when building the content of the label. This allows you to include dynamic information about the rating in the template.

**CSHTML**

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).ShowLabel(true).LabelTemplate("${value} Out Of 5").Render()
```

**LABELTEMPLATE.CS**

```
public ActionResult LabelTemplate()
{
 return View();
}
```

★ ★ ★ ☆ ☆ 3 Out Of 5

**Tooltip in ASP.NET MVC Rating Control**

The ASP.NET MVC rating control supports tooltip to show additional information in rating items by setting the [ShowTooltip](#) property. If enabled, the tooltip appears when the user hovers over a rating item.

**CSHTML**

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).ShowTooltip(true).Render()
```

**SHOWTOOLTIP.CS**

```
public ActionResult ShowTooltip()
{
 return View();
}
```



## Tooltip template

You can use the [TooltipTemplate](#) tag directive to specify a custom template for the **Tooltip** of the rating. The current value of the rating will be passed as the **value** property in the template context when building the content of the tooltip. This allows you to include dynamic information about the rating in the template.

**CSHTML**

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(4).TooltipTemplate("#template").Render()
<script id="template" type="text/x-jsrender">
 ${if(value==1)}
 Angry${else if(value==2)}
 Sad${else if(value==3)}
 Neutral${else if(value==4)}
 Good${else}
 Happy
 ${/if}
</script>
```

**TOOLTIPTEMPLATE.CS**

```
public ActionResult TooltipTemplate()
{
 return View();
}
```



## Tooltip customization

You can customize the appearance of the tooltips using the **CssClass** property of the ASP.NET MVC Rating control and by defining the custom styles for tooltip elements like the below example.



**Note:** You can find more information about customizing the appearance of the tooltip in the [Tooltip Customization](#) documentation.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).CssClass("customtooltip").Render()
<style>
 /* To change the radius of the tooltip corners. */
 .customtooltip.e-tooltip-wrap {
 border-radius: 3px;
 }
 /* To change the size of the tooltip content. */
 .customtooltip.e-tooltip-wrap .e-tip-content {
 font-size:14px;
 }
 /* To change the border color and width for tooltip. */
 .customtooltip.e-tooltip-wrap.e-popup {
 border: 2px solid #969393;
 }
 /* To change the color for arrow of the tooltip. */
 .customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
 border: 12px solid #9693
 }
 /* To change the top border color for arrow of the tooltip. */
 .customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
 border-top: 12.5px solid #969393;
 }
</style>
```

### CUSTOMTOOLTIP.CS

```
public ActionResult CustomTooltip()
{
 return View();
}
```



### Selection in ASP.NET MVC Rating Control

The ASP.NET MVC Rating control allows users to rate something using a visual scale, and the selection state can be changed by the user clicking or tapping on the stars in the rating scale or through code. The Rating control has a minimum value and a reset button, and provides customization options for the selected rating value and selection behavior.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).Render()
```

### SELECTION.CS

```
public ActionResult Selection()
{
 return View();
}
```



#### Min value

You can use the [Min](#) property of the ASP.NET MVC Rating control to set the minimum possible rating value the user can select. If you set the `Min` property to 2, then you will not be able to select a rating lower than 2.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Min(2).Render()
```

### MIN.CS

```
public ActionResult Min()
{
 return View();
}
```



#### Single selection

You can use the [EnableSingleSelection](#) property of the ASP.NET MVC Rating control to select only one item at a time. When the `EnableSingleSelection` property is set to `true`, only the selected item will be considered to be in the selected state, while all other items will be unselected.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).EnableSingleSelection(true).Render()
```

### SINGLESELECTION.CS

```
public ActionResult SingleSelection()
{
 return View();
}
```



### Show or hide reset button

You can reset the rating value to its default by using the [AllowReset](#) property of the ASP.NET MVC Rating control. When the [AllowReset](#) property is set to `true`, a reset button will be shown that allows the user to reset the rating value to its default.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).AllowReset(true).Render()
```

#### ALLOWRESET.CS

```
public ActionResult AllowReset()
{
 return View();
}
```



### Templates in ASP.NET MVC Rating Control

The ASP.NET MVC Rating control allows you to customize the appearance of the rating items using templates. You can use templates to specify a custom layout for the rating items, which can include any content you want. This allows you to create a more customized and interactive rating experience for the user.

The rating control supports below templates for item customization.

- [EmptyTemplate](#)
- [FullTemplate](#)

#### Empty (unrated) symbol template

To customize the appearance of **unrated** items, you can use the [EmptyTemplate](#) tag directive. It allows you to specify the desired custom content for the unrated items.

The `value` and `index` are available in the template context for accessing information about the un-rated item.

If the [FullTemplate](#) is not defined, the [EmptyTemplate](#) will be used as the default for both rated and unrated items. You can apply custom styles to differentiate between the rated and unrated states of the items.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).EmptyTemplate("<span class='custom-
font sf-rating-heart'>").Render()
<style>
.e-rating-container .custom-font {
```

```

/* To add the background color for the font icon. */
background: linear-gradient(to right, rgb(254,87,133,255) var(--
rating-value), transparent var(--rating-value));
/* To clip the background to the icon (text) alone. */
background-clip: text;
-webkit-background-clip: text;
/* To make the background color visible instead of font color. */
-webkit-text-fill-color: transparent;
/* To provide a border for font icon. */
-webkit-text-stroke: 1px rgb(254,87,133,255);
}
@@font-face {
 font-family: 'rating';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjluSfQAAAEoAAAAVmNtYXNlEudaAAABjAAAADhnbHlm4Li
FsgAAAcwAAAJsAGVhZCKCSVKAADQAAAAANmhoZWEIUQQAEEAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQCMATYAAAEAAAAACG1heHABDwCZAAABCAAAACBuYw1l75Kp8wAABDgAAAIzcG9zdDjyU90
AAAZUAAAAANwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAwABAAAAQAA2T6Kh18
PPPUACwQAAAAAAN+4AkEAAAAA37gCQQAAAAAD9APaAAAACAACAAAAAAAAAAAAEAAAADAI0AAgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wHnAgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAA
EAAAAAAAAAgAAAMAAAAUAAMAAQAAABQABAakAAAABAAEAAEAOCc//8AAOCb//8AAAABAAQAAAA
CAEAAAAAAAAIwBNgABAAAAAPZA9oAfAAAEw8WFR8PPw41Lx4jDwwvDw8GqAwMDAsKCGoJCAgIBwY
GBQUEBAMCAgEBAQECAwMEBQULFSMhOVJliOxTOSedFg0IBQQDAwIBAQEBAgIDBAQFBQYGBwgICak
KCgoLDawMDawMDQwMDQwZGBgYFxFVFBIRCAgGBwKLCwWNDg4PEBAQEREREhEODg4ODg4NA8IGBwc
ICakJCgoKCwsMCwWND40MDQ0ODQ0ODQ0ODQ0NDRUIMctEX26P/V5FKYcjFhQNDQ0ODQ0ODQ0NDgw
NDQWNCwWMCwoLCgoJCAkIBwcGBQUEAwMCAQECEBQYJCw4PERMKCGsMEQ8PDQ0LCwoICAYFBAMCAQE
BAgIEBAUAAGAAAAAD9APFAAMAJAAANzMRIwEPaXUXDwwRMzcfBDcXPwo9AS8FPwsvCDc1Pwg1LwU
1Pw01LwkHJT8ENS8LIw8BDK2tAfKCCgQBAQEBCERERITigkJKBAGIQc1Bx45k9sOBQgLDQsJBQM
EAgIECQYCAQEBAw4ECQgGBwMDAQEBAQMDAwkCAQEDFgsFBAQDAwICAgQECgEBAQKbwcGBQUEAwM
BAQEBAUHCQYUFBQYR/q0PCQQDAgEBAwMKDBUDbwYMCw0HB1oBhwHeAQUDA3YfCgQsOh0bHBovCQg
bDP6KAQEfAwEBAQIBAQMGCGoMBggICAUICQgLBQQEBAUDBgMHCAGMCACIBwYGBgUFCQQCBgIEDAk
GBQYHCQkKQgIBwsEAgUDAgQEBAUFBgCHCAcGBgYGCgkIBgICAQEBAUYxGRobDQ0MDQsiHjEEBAI
EAQECAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAAAEABgABAAEAAAAAAAAIABwAHAAEAAAAAAAAAMABgA
OAAEAAAAAAAAAQABgAUAAEAAAAAAAAUACwAAAEAAAAAAAAAYABgAlAAEAAAAAAAAoALAArAAEAAAAAAs
AEgBXAAMAAQQJAAAAAgBpAAMAAQQJAAEADABrAAMAAQQJAAIADgB3AAMAAQQJAAAMADACFAAMAAQQ
JAAQADACRAAMAAQQJAAUAFgCdAAMAAQQJAAAYADACzAAMAAQQJAAoAWAC/AAMAAQQJJAASAJAEXIHJ
hdGluZ1JlZ3VsYXJyYXRpbmdyYXRpbmdWZXJzaW9uIDEuMHJhdGluZ0ZvbnQgZ2VuZXJhdGVkIHV
zaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAACgBhAHQAaQB
uAGCAUgBlAGCAQBsAGEACgByAGEADABpAG4AZwByAGEADABpAG4AZwBWAGUACgBzAGkAbwBuACA
AMQAuADAACgBhAHQAaQBAGCAgBvAG4AdAAgAGCAZQBAGUACgBhAHQAZQBkACAADQBzAGkAbgB
nACAAUwB5AG4AYwBmAHUACwBpAG8AbgAgAE0AZQB0AHIAbwAgAFMADAB1AGQAaQBvAHcAdwB3AC4
AcwB5AG4AYwBmAHUACwBpAG8AbgAuAGMABwBtAAAAAIAAAAAAAAAACgAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAwECAQMBBAFAgVhcnQfGdh1bWIAAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-rating-"], [class*=" sf-rating-"] {
 font-family: 'rating' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;

```

```

 }
 .sf-rating-heart:before {
 content: "\e702";
 }
}
</style>

```

### EMPTYTEMPLATE.CS

```

public ActionResult EmptyTemplate()
{
 return View();
}

```



**Note:** The current value of the rating item available in the template context as `value` and in the rating item element as CSS Variable (`--rating-value`) can be used to support precision in templates.

### Full (rated) symbol template

To customize the appearance of **rated** items in the Syncfusion ASP.NET MVC rating control, you can use the `FullTemplate` tag directive. This directive allows you to specify a custom layout for the rated items, which can include any content you desire.

The `value` and `index` are available in the template context for accessing information about the rated item.

### CSHTML

```

@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).EmptyTemplate("").FullTemplate("").Render()
<style>
 .e-rating-container .custom-font {
 /* To change the icon font color. */
 color: rgb(255,215,0);
 }
 @@font-face {
 font-family: 'rating-template';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAKAIAAAwAgTlMvMjltSfMAAAEoAAAAVmNtYXNlEODaAAABjAAAADhnbHlm+icDjQAAAcwAAAE0aGVhZCK49ucAAADQAAAAANmhoZWEIUQQEAAAArAAAACRobXR4DAAAAAAAAAYAAAAAMbG9jYQAcAJ0AAAEHEAAAACG1heHABDWBkAAABCAAAACBuYW11mYExxgAAAwAAAAKFcG9zdCH169QAAAWIAAAQAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAwABAAAAQAAGPX4jF8PPPUACwQAAAAAN/TWPsAAAAA39NY+wAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAADAFgAAgAAAAAAGAAAAoACgAAAP8AAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQQAAAAAXAQAAAAAAAAABAAAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAAUAAMAAQAAABQABAakAAAABAAEAAEAOCB//8AAOCa//8AAAABAAQAAAA
BAAIAAAAAABwAmgABAAAAAAP0A/QACQAAAQUTAYUFAxMlAwFn/qX6OwE1ATU7+v6lmQKsNf7//pa
srAFqAQE1AUgAAAIAAAAAAAA/QD5AAAdAFcAAAEfBAUPAxUTLwEjDwETNS8DJT8EJwMFDwQVHwIDBx8
EMzclBRczPwU1Az8CNS8DJQMvBisBDwUCYAICBgMHASCwBAMCGuoHCAjPggIDBLEBHgcGBgJiHXb
+uQgHBgQBAGTUHgECBAUHCakIAQ4BDgcJBAQEbwQDHDMEAgMFBgf+tHYDAgMEBAQEBAQEBAQEAWI
CnwMDBgIDNBAGBgYE/uCBAGKBAR0HBgYGsDQCBAyD3Fr+9jwDBAcHBQgIB9T+twUIBwcEAWKVlQI

```

```

BAGIFBwgJAUnUBwgJCAcFBD0BCgQDBAICAgEBAgICBAMAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAA
AAAEADwABAAEAAAAAAAAIABwAQAAEAAAAAAAAAMADwAXAAEAAAAAAAAQADwAmAAEAAAAAAAAUACwA1AAE
AAAAAAAAAYADwBAAEAAAAAAAAoALABPAEAAAAAAAAsAEgB7AAMAAQQJAAAAAgCNAAMAAQQJAAEAHgC
PAAMAAQQJAAIADgCtAAMAAQQJAAAMAHgC7AAMAAQQJAAQAHgDZAAMAAQQJAAUAFgD3AAMAAQQJAAAY
AHgENAAMAAQQJAAoAWAERAAAMAAQQJAAAsAJAGDIHJhdGluZy10ZW1wbGF0ZVJlZ3VsYXJyYXRpbmc
tdGVtcGxhdGVyYXRpbmctdGVtcGxhdGVWZXJzaW9uIDEuMHJhdGluZy10ZW1wbGF0ZUZvbnQgZ2V
uZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACA
AcgBhAHQAaQBwAGcALQB0AGUAbQBwAGwAYQB0AGUAUgBlAGcAdQBsAGEAcgByAGEAdABpAG4AZwA
tAHQAZQBtAHAAbABhAHQAZQBByAGEAdABpAG4AZwAtAHQAZQBtAHAAbABhAHQAZQBWAGUAcgBzAGk
AbwBuACAAMQAuADAACgBhAHQAaQBwAGcALQB0AGUAbQBwAGwAYQB0AGUARgBvAG4AdAAgAGcAZQB
uAGUAACgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAACwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAACwBpAG8AbgAgAE0AZQB0AHI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAwECAQMBAAJZmlsbC1zdGFyCmVtcHR5LXN
0YXIAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'rating-template' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-fill-star:before {
 content: "\e700";
}
.sf-icon-empty-star:before {
 content: "\e701";
}
</style>

```

### FULLTEMPLATE.CS

```

public ActionResult FullTemplate()
{
 return View();
}

```



### Using Emoji icon as rating symbol

You can use emojis of your choice as rating symbol by specifying them as template content within the `EmptyTemplate` tag directive.

### CSHTML

```

@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(4).EnableAnimation(false).EnableSingleSel
ection(true).EmptyTemplate("#template").Render()

```

```
<script id="template" type="text/x-jsrender">
 ${if(index==0)}
 😡${else if(index==1)}
 🙁${else if(index==2)}
 😐${else if(index==3)}
 🙂${else}
 😀
 ${/if}
</script>
<style>
 /* To change the color of an unselected rating item. */
 .e-rating-item-container:not(.e-rating-selected) .emoji {
 filter: grayscale(1);
 }
</style>
```

### EMOJIICON.CS

```
public ActionResult EmojiIcon()
{
 return View();
}
```



### Using SVG icon as rating symbol

You can use SVG icons of your choice as rating symbol by specifying them as template content within the `EmptyTemplate` and `FullTemplate` tag directives.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(4).EnableAnimation(false).EmptyTemplate("#emptyTemplate").FullTemplate("#fullTemplate").Render()
<script type="text/x-jsrender" id="fullTemplate">
 <svg width="35" height="25" class="e-rating-svg-icon">
 <defs>
 <linearGradient id="grad${index}" x1="0%" y1="0%" x2="100%"
y2="0%">
 <stop class="start" offset="0%" />
 <stop class="end" offset="100%" />
 </linearGradient>
 </defs>
 <rect width="35" height="25" fill="url(#grad${index})"
style="stroke-width:2;stroke:rgb(173,181,189)" />
 </svg>
</script>
<script type="text/x-jsrender" id="emptyTemplate">
 <svg width="35" height="25" class="e-rating-svg-icon">
 <rect width="35" height="25" fill="transparent" style="stroke-
width:2;stroke:rgb(173,181,189)" />
 </svg>
</script>
```

```

<style>
/* To change the size between items */
.e-rating-container .e-rating-item-container {
 padding: 0px;
}
/* To set the gradient color */
.e-rating-svg-icon #grad0 .start {
 stop-color: #FF0000;
}
.e-rating-svg-icon #grad0 .end,
.e-rating-svg-icon #grad1 .start {
 stop-color: #ff5101;
}
.e-rating-svg-icon #grad1 .end,
.e-rating-svg-icon #grad2 .start {
 stop-color: #ffc801;
}
.e-rating-svg-icon #grad2 .end,
.e-rating-svg-icon #grad3 .start {
 stop-color: #dbe300;
}
.e-rating-svg-icon #grad3 .end,
.e-rating-svg-icon #grad4 .start {
 stop-color: #8bc301;
}
.e-rating-svg-icon #grad4 .end {
 stop-color: #4eaa01;
}
</style>

```

### SVGTEMPLATE.CS

```

public ActionResult SVGTemplate()
{
 return View();
}

```



### Using PNG image as rating symbol

You can use PNG images of your choice as rating symbol by specifying them as template content within the `EmptyTemplate` and `FullTemplate` tag directives.

### CSHTML

```

@using Syncfusion.EJ2.Inputs
@{
 var fulltemplate = ""; //Provide the URL for the image here.
 var emptytemplate = ""; //Provide the URL for the image here.
}
@Html.EJS().Rating("rating").Value(3).EmptyTemplate("#emptytemplate").FullTemplate("#fulltemplate").Render()
<script id="fulltemplate" type="text/x-jsrender">

</script>

```



```

</script>
<script id="emptytemplate" type="text/x-jsrender">

</script>

```

### PNGIMAGE.CS

```

public ActionResult PNGImage()
{
 return View();
}

```



### Events in Rating Control

This section describes the rating events that will be triggered when appropriate actions are performed. The following events are available in the rating control.

#### BeforeItemRender

The rating control triggers the [BeforeItemRender](#) event before rendering each rating item. The `RatingItemEventArgs` passed as an event argument provides the details of the item to be rendered.

### CSHTML

```

@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").BeforeItemRender("BeforeItemRender").Render()
<script>
 function BeforeItemRender(args) {
 // Here, you can customize your code.
 }
</script>

```

### BEFOREITEMRENDER.CS

```

public ActionResult BeforeItemRender()
{
 return View();
}

```

#### Created

The rating control triggers the [Created](#) event when the rendering of the rating control is completed.

### CSHTML

```

@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Created("Created").Render()
<script>
 function Created(args) {
 // Here, you can customize your code.
 }
</script>

```

**CREATEDEVENT.CS**

```
public ActionResult CreatedEvent()
{
 return View();
}
```

**ValueChanged**

The rating control triggers the [ValueChanged](#) event when the value of the rating is changed. The `RatingChangedEventArgs` passed as an event argument provides the details when value is changed.

**CSHTML**

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").ValueChanged("ValueChanged").Render()
<script>
 function ValueChanged(args) {
 // Here, you can customize your code.
 }
</script>
```

**VALUECHANGED.CS**

```
public ActionResult ValueChanged()
{
 return View();
}
```

**OnItemHover**

The rating control triggers the [OnItemHover](#) event when the rating item is hovered. The `RatingHoverEventArgs` passed as an event argument provides the details of the hovered item.

**CSHTML**

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").OnItemHover("OnItemHover").Render()
<script>
 function OnItemHover(args) {
 // Here, you can customize your code.
 }
</script>
```

**ITEMHOVER.CS**

```
public ActionResult ItemHover()
{
 return View();
}
```

**Appearance in ASP.NET MVC Rating Control**

You can also customize the appearance of rating control.

### Items count

You can specify the number of rating items using the [ItemsCount](#) property.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).ItemsCount(8).Render()
```

#### ITEMSCOUNT.CS

```
public ActionResult ItemsCount()
{
 return View();
}
```



### Disabled

You can disable the Syncfusion ASP.NET MVC Rating control by using the [Disabled](#) property. When the **Disabled** property is set to **true**, the rating control will be disabled and the user will not be able to interact with it and a disabled rating control may have a different visual appearance than an enabled one.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).Disabled(true).Render()
```

#### DISABLED.CS

```
public ActionResult Disabled()
{
 return View();
}
```



### Visible

You can use the [Visible](#) property of the ASP.NET MVC Rating control to control the visibility of the control. When the **Visible** property is set to **true**, the rating control will be visible on the page. When it is set to **false**, the control will be hidden.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).Visible(true).Render()
```

#### VISIBLE.CS

```
public ActionResult Visible()
```

```
{
 return View();
}
```



### Read only

You can use the [ReadOnly](#) property of the ASP.NET MVC Rating control to make the control non-interactive and prevent the user from changing the rating value.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).ReadOnly(true).Render()
```

### READONLY.CS

```
public ActionResult ReadOnly()
{
 return View();
}
```



### CssClass

You can customize the appearance of the rating control, such as by changing its colors, fonts, sizes, or other visual aspects by using the [CssClass](#) property.

#### Changing rating symbol border color

You can change the rating icon border color in ASP.NET MVC Rating control, you can use the [CssClass](#) property and set the [text-stroke](#) CSS property of [.e-rating-icon](#) to your desired border color.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).CssClass("custom-font").Render()
<style>
 .e-rating-container.custom-font .e-rating-item-list:hover .e-rating-
 item-container .e-rating-icon,
 .e-rating-container.custom-font .e-rating-item-container .e-rating-icon
 {
 /* To change rating symbol border color */
 -webkit-text-stroke: 2px #ae9e9d;
 }
</style>
```

### BORDERCOLOR.CS

```
public ActionResult BorderColor()
```

```
{
 return View();
}
```



#### Changing rated/un-rated symbol fill color

You can customize the fill colors of rated and un-rated icons in the Rating control using the `CssClass` property and the `linear-gradient` color-stops in the `background` CSS property of `.e-rating-icon`. The **first** color-stop defines the rated fill color and the **second** defines the un-rated fill color.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).CssClass("custom-font").Render()
<style>
 .e-rating-container.custom-font .e-rating-item-list:hover .e-rating-
 item-container .e-rating-icon,
 .e-rating-container.custom-font .e-rating-item-container .e-rating-icon
 {
 /* To change rated symbol fill color and un-rated symbol fill color
 */
 background: linear-gradient(to right, #ffe814 var(--rating-value),
 #d8d7d4 var(--rating-value));
 background-clip: text;
 -webkit-background-clip: text;
 }
</style>
```

#### FILLCOLOR.CS

```
public ActionResult FillColor()
{
 return View();
}
```

This will customize the rated fill color to `#ffe814` and un-rated fill color to `#d8d7d4`. `--rating-value` in the `linear-gradient` provides the current value of the rating item.



#### Changing the item spacing

You can change the space between the rating items in ASP.NET MVC Rating control, by using the `CssClass` property and setting the `margin` / `padding` CSS property of `.e-rating-item-container` to your desired size.

#### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).CssClass("custom-font").Render()
<style>
```

```
.e-rating-container.custom-font .e-rating-item-container {
 /* To change the size between items */
 margin: 0px 7px;
}
</style>
```

### ITEMSSPACING.CS

```
public ActionResult ItemSpacing()
{
 return View();
}
```



### Changing icon using CssClass

You can change the rating item icon in ASP.NET MVC Rating control, you can use the `CssClass` property and set the `content` CSS property of `.e-icons.e-star-filled:before` to your desired font icon.

### CSHTML

```
@using Syncfusion.EJ2.Inputs
@Html.EJS().Rating("rating").Value(3).CssClass("custom-icon").Render()
<style>
 .custom-icon .e-icons.e-star-filled:before {
 content: "\e702";
 }
 @@font-face {
 font-family: 'custom-icon';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAKAIAAAwAgT1MvMjlvSfQAAAEoAAAAVmNtYXNlEudXAAABiAAAADZnbHlmVIZ
rowAAACgAAAEYAGVhZCK6KOUAAADQAAAAANmhoZWEIUAQDAAAArAAAACRobXR4CAAAAAAAAAAYAAAA
IbG9jYQCMAAAAAAAAAAAAABmlheHABDQCJAAABCAAAACBuYW1lv3dY+QAAAUAAAAJVCg9zdN12Ynk
AAAU4AAAAALwABAAAAEAAAAAFwEAAAAAAAAAD8wABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAEEGKKhV8
PPPUACwQAAAAAN/UcgCAAAAAA39RyBwAAAAAD8wPaAAAAACAACAAAAAAAAAAAAEAAAACAH0AAQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnAgQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAQA
AAAAACAAAAAwAAABQAawABAAAAFAAEACIAAAAEAAQAAQAA5wL//wAA5wL//wAAAAEABAAAAEAAAA
AAAAAJAAAAEAAAAAAAA/MD2gB8AAATDxYVHw8/DjUvHiMPDC8PDwaoDAwMCwoKCgkICAgHBgYFBQQ
EAwICAQEBAQIDAwQFBQsVIyE5UmWI7FM5IR0WDQgFBAMDAGEBQECAGMEBAUFBgYHCAgICQoKCgs
MDAwMDAwNDawNDBkYGBgXFRUUEhEICAYHCQsLDA0ODg8QEBARERESEQ4ODg4ODg0DwgYHBwgICQk
KCgoLCwwLDA0MDQwNDQ4NDQ4NDQ4NDQ0NFSIwK0Rfbo/9XkUrJyMWFA0NDQ4NDQ4NDQ0ODA0NDA0
LDawLCgsKCgkICQgHBwYFBQQDAwIBAQIFBgkLDg8REwoKCwwRDw8NDQsLCggIBgUEAwIBAQECAGQ
EBQAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAAsAAQABAAAAAAACAAcADAABAAAAAADAAAsAEwA
BAAAAAAEAAsAHgABAAAAAAFAAsAKQABAAAAAAAGAAsANAABAAAAAAAKACwAPwABAAAAAALABI
AawADAAEEECQAAAAIAfQADAAEEECQABABYafwADAAEEECQACAA4AlQADAAEEECQADABYAowADAAEEECQA
EABYAuQADAAEEECQAFABYazwADAAEEECQAGABYA5QADAAEEECQAKAFgA+wADAAEEECQALACQBUyBjdXN
0b20taWNvblJlZ3VsYXJjdXN0b20taWNvbml3RvblpY29uVmVyc2lvbiAxLjBjdXN0b20taWN
vbKZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXN
pb24uY29tACAAYwB1AHMAAdABvAG0ALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAAYwB1AHMAAdABvAG0
ALQBpAGMABwBuAGMAdQBzAHQAbwBtAC0AaQBjAG8AbgBWAGUAcgBzAGkAbwBuACAAMQAuADAAYwB
1AHMAAdABvAG0ALQBpAGMABwBuAEYAbwBuAHQAIAIBnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAACwBpAG4
AZwAgAFMAeQBwAGMAZgB1AHMAAQBVAG4AIAIBNAGUAdABYAG8AIAIBTAHQAdQBkAGkAbwB3AHcAdwA
```

```

uAHMAeQBuaGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAIBAgEDAAVoZWYdAAAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
}
.custom-icon .e-icons.e-star-filled {
 font-family: 'custom-icon' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
</style>

```

### CHANGEICON.CS

```

public ActionResult ChangeIcon()
{
 return View();
}

```



### Accessibility in ASP.NET MVC Rating Control

Accessibility is achieved in the rating control through **WAI-ARIA** standard and keyboard navigations. The Rating control can be effectively accessed through assistive technologies such as screen readers.

#### Keyboard interaction

The rating control is interactive with below keyboard shortcuts.

| Keyboard shortcuts | Actions |

|-----|-----|

| Space | If a **Reset Button** is focused, resets the value to **Min** value. |

| ArrowUp | Increases the value. |

| ArrowLeft | Decreases the value and in RTL mode, increases the value. |

| ArrowDown | Decreases the value. |

| ArrowRight | Increases the value and in RTL mode, decreases the value. |

#### ARIA attribute

The following ARIA attributes are used in rating control based on its state.

| Properties | Functionality |

| ----- | ----- |

| role | This attribute is added to the div element to describe the actual role. |

| aria-label | Attribute provides the text label with some default description for the Rating and its items.  
|

| aria-valuemin | It defines the minimum value of rating. |

| aria-valuemax | It defines the maximum value of rating. |

| aria-valuenow | It defines the current value of rating. |

## Ribbon

### Getting Started with ASP.NET MVC Ribbon Control

This section briefly explains about how to include **ASP.NET MVC Ribbon** control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,



### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ \_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Ribbon control

Now, add the Syncfusion ASP.NET MVC Ribbon control in `~/Home/Index.cshtml` page.

#### INDEX.CSHTML

```
@Html.EJS().Ribbon("ribbon").Render()
```

### Adding Ribbon Tab

In Ribbon, the options are arranged in tabs for easy access. You can use the `Tabs` property of ribbon to define the ribbon tab like below.

#### INDEX.CSHTML

```
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Add();
}).Render()
```

### Adding Ribbon Group

To define a ribbon group under each tab, you can use the **Groups** property of ribbon tab like below. The **Orientation** property of ribbon group defines whether the collection of items will be rendered column-wise or row-wise.

#### INDEX.CSHTML

```
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(groups =>
 {
 groups.Header("Clipboard").Orientation(ItemOrientation.Row).Add();
 }).Add();
}).Render()
```

### Adding Ribbon Items

You can use the **Collections** property of ribbon group to define each ribbon collection that contains one or more items. To define each ribbon item, you can use the **Items** property of ribbon collection and the **Type** property of ribbon item to specify the type of control to be rendered, like a button, a drop-down button, a combo box, and more.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
 "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
 { Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(groups =
 {
 groups.Header("Clipboard").Orientation(ItemOrientation.Row).Collections(coll
 ection =>
 {
 collection.Id("paste-collection").Items(items =>
 {
 items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitbutton =>
 {
 splitbutton.IconCss("e-icons e-paste").Items(pasteOptions).Content("Paste");
 }).Add();
 }).Add();
 collection.Id("cutcopy-collection").Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }
 }
}).Add();
```

```

 }).Add();
 }).Add();
 }).Render();

```

The following example illustrates how tabs, groups, collections, and items are used in a ribbon control to form the ribbon layout.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text
= "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem {
Text = "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };

 List<string> fontSize = new List<string>() { "8", "9", "10", "11", "12",
"14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72", "96" };
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
 List<MenuItem> fileOptions = new List<MenuItem>()
 {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new",
Id="new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open",
Id="open" },
 new MenuItem { Text = "Rename", IconCss = "e-icons e-rename",
Id="rename" },
 new MenuItem { Text = "Save as", IconCss = "e-icons e-save",
Id="save" }
 };
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Text("File").Visible(true).MenuItems(fileOptions);
}).Tabs(tab => {
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").ShowLauncherIcon(true).GroupIconCss("e-
icons e-paste").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitbutton =>
 {
 splitbutton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste");
 }).Add();
 }).Add();
 }).Add();
 }
}

```

```

collection.Items(items =>
{
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
}).Add();

group.Header("Font").EnableGroupOverflow(true).Orientation(ItemOrientation.Row).GroupIconCss("e-icons e-bold").CssClass("font-group").Collections(collections =>
{
 collections.Id("ribbon-collection").Items(items =>
 {
 items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
 {
 comboBox.DataSource(fontStyle).Index(3).AllowFiltering(true).Width("150px");
 }).Add();

 items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
 {
 comboBox.DataSource(fontSize).Index(3).Width("65px");
 }).Add();
 }).Add();
 collections.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).AllowedSizes(RibbonItemSize.Small).DisplayOptions(Syncfusion.EJ2.Ribbon.DisplayMode.Simplified).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456");
 }).Add();

 items.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Small).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").IsToggle(true);
 }).Add();

 items.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Small).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").IsToggle(true);
 }).Add();
 });
});

```

```

items.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Small).ButtonSettings(button =>
{
 button.IconCss("e-icons e-underline").IsToggle(true);
}).Add();

items.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Small).ButtonSettings(button =>
{
 button.IconCss("e-icons e-strikethrough").IsToggle(true);
}).Add();

items.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Small).ButtonSettings(button =>
{
 button.IconCss("e-icons e-change-case").IsToggle(true);
}).Add();
}).Add();
group.Header("Editor").GroupIconCss("e-icons e-edit").Collections(collections =>
{
 collections.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.Content("Editor").IconCss("e-icons e-edit");
 }).Add();
 }).Add();
}).Add();
tab.Header("Insert").Groups(groups =>
{
 groups.Header("Tables").Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Id("defaultitem18").Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-table").Content("Table").Items(tableOptions);
 }).Add();
 }).Add();
 }).Add();

 groups.Header("Illustrations").EnableGroupOverflow(true).Orientation(ItemOrientation.Row).GroupIconCss("e-icons e-image").Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Id("defaultitem23").Type(RibbonItemType.Button).ButtonSettings(button =>
 {

```

```

 {
 button.IconCss("e-icons e-chart").Content("Chart");
 }).Add();
 }).Add();
}).Add();
groups.Header("Media").Collections(collections =>
{
 collections.Items(items =>
 {
 items.Type(RibbonItemType.Template).ItemTemplate("<span class='e-icons e-
video'>Video").Add();
 }).Add();
 }).Add();
}).Add();
tab.Header("View").Groups(groups =>
{
 groups.Header("Views").GroupIconCss("e-icons e-
print").Orientation(ItemOrientation.Row).Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-print-layout").Content("Print
Layout");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-web-layout").Content("Web
Layout");
 }).Add();
 }).Add();
 }).Add();
 groups.Header("Show").Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
 {
 checkBox.Label("Ruler").Checked(false);
 }).Add();

 items.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
 {
 checkBox.Label("Gridlines").Checked(false);
 }).Add();

 items.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
 {
 checkBox.Label("Navigation Pane").Checked(true);
 }).Add();
 }).Add();
 }).Add();
}).Render()

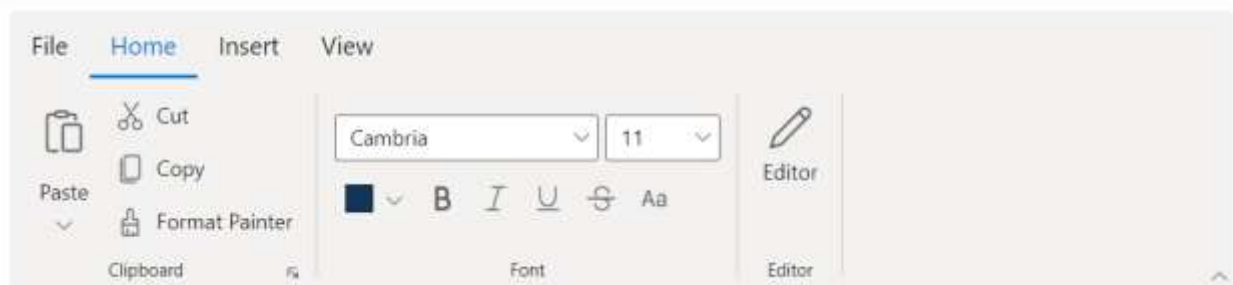
```

```

<style>
 .ribbonTemplate {
 display: flex;
 align-items: center;
 justify-content: center;
 cursor: pointer;
 }
 .ribbonTemplate.Large {
 flex-direction: column;
 }
 .ribbonTemplate.Large .e-icons {
 font-size: 35px;
 }
 .ribbonTemplate.Medium .e-icons,
 .ribbonTemplate.Small .e-icons {
 font-size: 20px;
 margin: 15px 5px;
 }
 .ribbonTemplate.Small .text {
 display: none;
 }
 .font-group .e-ribbon-group-content {
 justify-content: center;
 }
</style>

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Ribbon control will be rendered in the default web browser.



## Tabs and Groups

The Ribbon control consists of a series of tabs that are organized into groups to enable quick access to specific commands or tools. Each tab contains a set of groups, and each group contains collections of items that are logically related to each other.

### Adding Tabs

You can use the [Tabs](#) property to add tabs to the Ribbon control and define the content of the tab header by using the [Header](#) property.

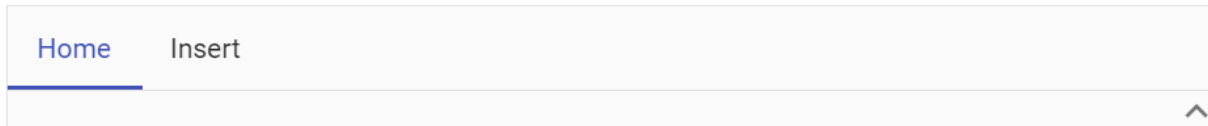
### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{

```

```
tab.Header("Home").Add();
}).Render()
```

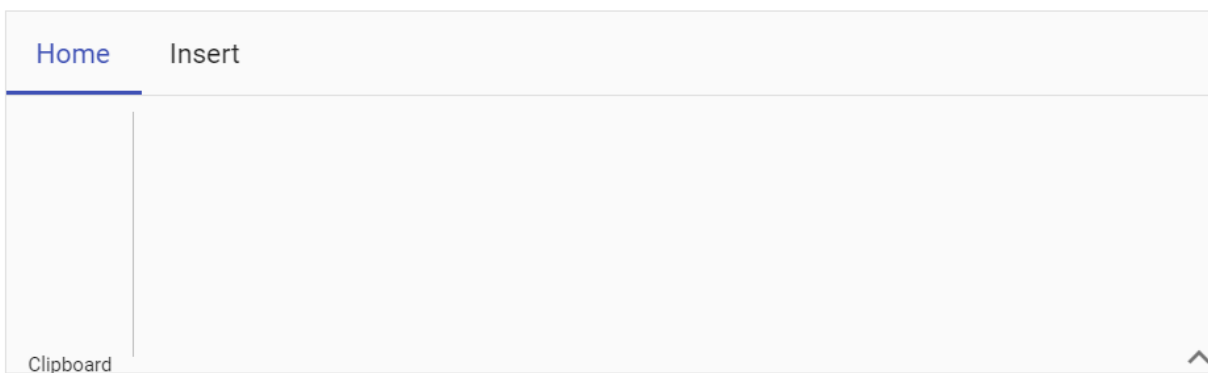


### Adding Groups

You can use the [Groups](#) property to add groups for each tab in the Ribbon and define the name of the group header by using the [Header](#) property.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(groups =>
 {
 groups.Header("Clipboard").Add();
 }).Add();
 tab.Header("Insert").Groups(groups =>
 {
 groups.Header("Tables").Add();
 }).Add();
}).Render()
```



### Adding Items

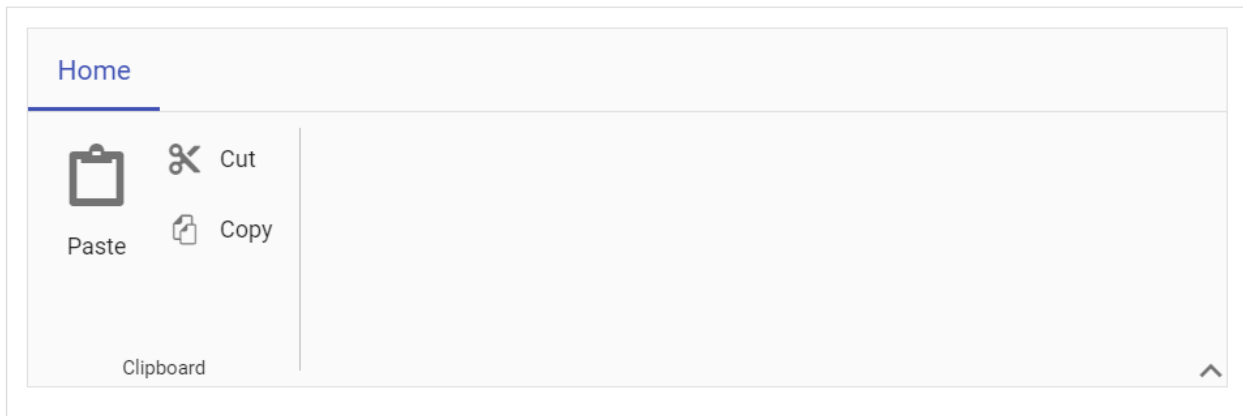
You can add collections of items to each group by using the [Collections](#) and [Items](#) properties.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
```



```
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```



For more information on the built-in and how to add custom Ribbon items, you can visit the [items](#) page.

### Ribbon Items

Ribbon renders various built-in items based on the item [Type](#) property. By default, the type property is set as **Button** which renders the Button.

### Built-in items

You can render the built-in Ribbon items by using the [Items](#) property, to specify the [Type](#) property.

The following table explains the built-in items and their actions.

| Built-in Ribbon Items | Actions |
|-----------------------|---------|
| -----                 | -----   |

- | Button | Renders button as ribbon item.
- | CheckBox | Renders checkbox as ribbon item.
- | DropDown | Renders dropdownbutton as ribbon item.
- | SplitButton | Renders splitbutton as ribbon item.
- | ComboBox | Renders combobox as ribbon item.
- | ColorPicker | Renders color picker as ribbon item.
- | GroupButton | Renders groupbutton as ribbon item.

### Button items

You can render the built-in button Ribbon item by setting the [Type](#) property as `Button`. You can also customize the button item using the [RibbonButtonSettings](#), which provides options such as `iconCss`, `content`, `isToggle` and more.

### Toggle button

The [IsToggle](#) property can be used to define whether the button act as a toggle button or not. By default the value is `false`.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut").IsToggle(true);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render();
```

### CheckBox items

You can render the built-in checkBox Ribbon item by setting the [Type](#) property to `CheckBox`. You can also customize the checkBox item using the [RibbonCheckBoxSettings](#), which provides options such as `labelPosition`, `label`, `checked` and more.

### Checkbox state

You can use the [Checked](#) property to handle the checked or unchecked state. By default, the value is `false`.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
```

```
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("View").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
 {
 checkBox.Checked(true).Label("Ruler");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```

### Defining label

You can use the [Label](#) property to add a caption for the CheckBox. The label position can be set **Before** or **After**, by using the [LabelPosition](#) property. By default, the labelPosition is **After**.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("View").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
 {
 checkBox.Checked(true).Label("Ruler").LabelPosition("Before");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```

### DropDown button items

You can render the built-in dropDown Ribbon item by setting the [Type](#) property to **DropDown**. You can also customize the dropDown item through [RibbonDropDownSettings](#), which provides options such as **iconCss**, **content**, **target** and more.

### Target

The [Target](#) property specifies the element selector to be displayed in the DropDownButton popup.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
```

```

string[] pictureOptions = new string[] { "This device", "Stock Images",
"Online Images" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Illustrations").Collections(collection =>
{
collection.Items(item =>
{
item.Type(RibbonItemType.DropDown).DropDownSettings(dropdown =>
{
dropdown.IconCss("e-icons e-image").Content("Pictures").Target("#listView");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
@Html.EJS().ListView("listView").ShowHeader(true).HeaderTitle("Insert
Picture From").DataSource(pictureOptions).Render()

```

### Split button items

You can render the built-in splitButton Ribbon item by setting the [Type](#) property to `SplitButton` you can render a splitButton item. You can also customize the SplitButton item through [RibbonSplitButtonSettings](#), which provides options such as `IconCss`, `Items`, `Target` and more.

### Target

The [Target](#) property specifies the element selector to be displayed in the SplitButton popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
string[] pictureOptions = new string[] { "This device", "Stock Images",
"Online Images" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Illustrations").Collections(collection =>
{
collection.Items(item =>
{
item.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitbutton =>
{
splitbutton.IconCss("e-icons e-
image").Content("Pictures").Target("#listView");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()

```

```
@Html.EJS().ListView("listView").ShowHeader(true).HeaderTitle("Insert
Picture From").DataSource(pictureOptions).Render()
```

### Combobox items

You can render the built-in comboBox Ribbon item by setting the [Type](#) property to `ComboBox` you can render a comboBox item. You can also customize the ComboBox item through [RibbonComboBoxSettings](#), which provides options such as `AllowFiltering`, `AutoFill`, `Index`, `SortOrder` and more.

### Filtering

You can use the [AllowFiltering](#) property to filter the data items. The filtering operation is initiated automatically, as soon as you start typing characters. If no match is found, the value of the `noRecordsTemplate` property will be displayed. By default, the value is `false`.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
 "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("View").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(combobox =>
 {
 combobox.DataSource(fontStyle).AllowFiltering(true);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
```

### Index

You can use the [Index](#) property to get or set the selected item in the combobox.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
 "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("View").Collections(collection =>
```

```
{
collection.Items(item =>
{
item.Type(RibbonItemType.ComboBox).ComboBoxSettings(combobox =>
{
combobox.DataSource(fontStyle).Index(3);
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
```

### SortOrder

You can use the [SortOrder](#) property to specify the order in which the DataSource should be sorted.

|            |                                                |
|------------|------------------------------------------------|
| None       | The data source is not sorted.                 |
| Ascending  | The data source is sorted in ascending order.  |
| Descending | The data source is sorted in descending order. |

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("View").Collections(collection =>
{
collection.Items(item =>
{
item.Type(RibbonItemType.ComboBox).ComboBoxSettings(combobox =>
{
combobox.DataSource(fontStyle).SortOrder("Descending");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
```

### Colorpicker items

You can render the built-in colorPicker Ribbon item by setting the [Type](#) property to **ColorPicker**. You can also customize the colorPicker item through [RibbonColorPickerSettings](#), which provides options such as **Value**, **Columns**, **ShowButtons** and more.

### Value

You can use the [Value](#) property to specify the color value. The value should be specified as Hex code.

**INDEX.CSHTML**

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Inputs
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("View").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorpicker =>
 {
 colorpicker.Value("035a");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render();
 }
```

*Groupbutton items*

You can render the built-in groupbutton Ribbon item by setting the [Type](#) property to `GroupButton`. You can also customize the groupbutton item using the [RibbonGroupButtonSettings](#), which provides options such as `Selection` and `Items`.

*Items*

You can render the groupbutton items by using [Items](#) property. You can also customize the groupbutton items through [RibbonGroupButtonItem](#), which provides options such as `Content`, `IconCss`, `Selected` and more.

*Item content*

You can use the [Content](#) property to define the text content for the groupbutton.

**CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> groupButtonContent = new
 List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-left", Content
 = "Align Left" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-center", Content
 = "Align Center" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-right", Content
 = "Align Right" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-justify", Content =
 "Justify" }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
```

```

{
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Medium).GroupButtonSettings(groupButton =>
 {
 groupButton.Items(groupButtonContent);
 }).Add();
 }).Add();
 }).Add();
}).Render()

```



### Icon only

You can use the [IconCss](#) property to customize the groupbutton icon. If the `IconCss` property is not defined, the groupbutton will not be rendered.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> groupButtonIcon = new
 List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-left" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-center" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-right" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-justify" }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).GroupButtonSettings(groupButton =>
 {
 groupButton.Items(groupButtonIcon);
 }).Add();
 });
 });
 });
}).Render()

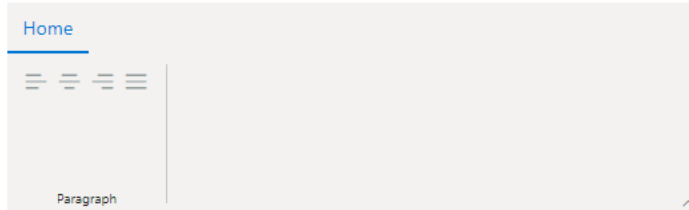
```



```

 }).Add();
 }).Add();
 }).Add();
}).Render();

```



### Selection

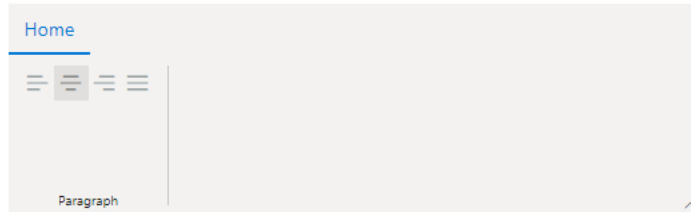
You can use the [Selected](#) property to select the groupbutton item initially. When set to `true`, the button will be selected. By default the `Selected` property is false.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> groupButtonSelected = new
 List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-left", Content
= "Align Left" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-center", Content
= "Align Center", Selected = true },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-right", Content
= "Align Right" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-justify", Content =
"Justify" }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).Gr
oupButtonSettings(groupButton =>
 {
 groupButton.Items(groupButtonSelected);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()

```



### Single selection

You can set the [Selection](#) property value as `RibbonGroupButtonSelection.Single` to make one selection at a time. It automatically deselects the previous choice when a different item is clicked.

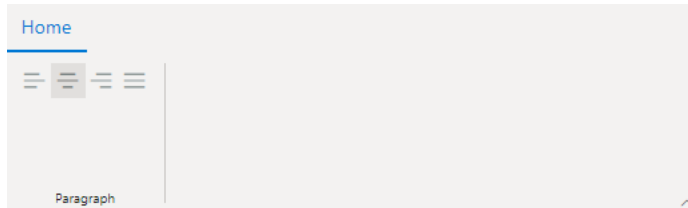
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> singleSelection = new
List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-left", Content
= "Align Left" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-center", Content
= "Align Center", Selected = true },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-right", Content
= "Align Right" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-justify", Content =
"Justify" }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {

items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).Gr
oupButtonSettings(groupButton =>
 {

groupButton.Selection(RibbonGroupButtonSelection.Single).Items(singleSelecti
on);

 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```



### Multiple selection

You can set the [Selection](#) property value as `RibbonGroupButtonSelection.Multiple` to select more than one button at a time. Users can select a button one by one to select multiple buttons.

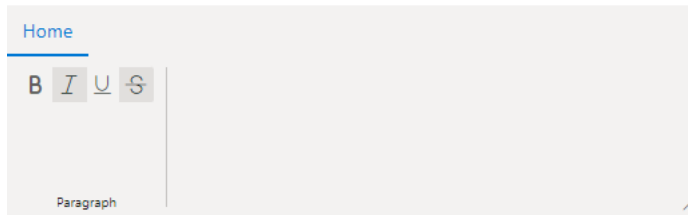
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> multipleSelection = new
List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-bold", Content =
"Bold" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-italic", Content =
"Italic", Selected = true },
 new RibbonGroupButtonItem { IconCss = "e-icons e-underline", Content =
"Underline" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-strikethrough",
Content = "Strikethrough", Selected = true }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {

items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).Gr
oupButtonSettings(groupButton =>
 {

groupButton.Selection(RibbonGroupButtonSelection.Multiple).Items(multipleSel
ection);

 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
}
```



### Groupbutton in simplified mode layout

In simplified mode, the groupbutton will be rendered as a dropdownbutton. The dropdownbutton icon will be updated based on the button item selected. The initial button icon will be the set, if none of the buttons are selected.

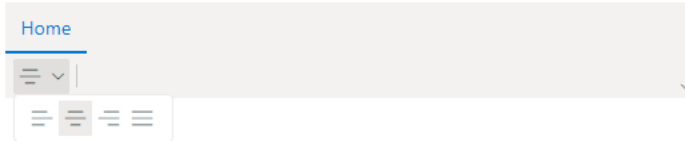
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> groupButtonSelected = new
List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-left", Content
= "Align Left" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-center", Content
= "Align Center", Selected = true },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-right", Content
= "Align Right" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-justify", Content =
"Justify" }
 };
}
@Html.EJS().Ribbon("ribbon").ActiveLayout(RibbonLayout.Simplified).Tabs(tab
=>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {

items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).Gr
oupButtonSettings(groupButton =>
 {

groupButton.Selection(RibbonGroupButtonSelection.Single).Items(groupButtonSe
lected);

 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 }
}
```

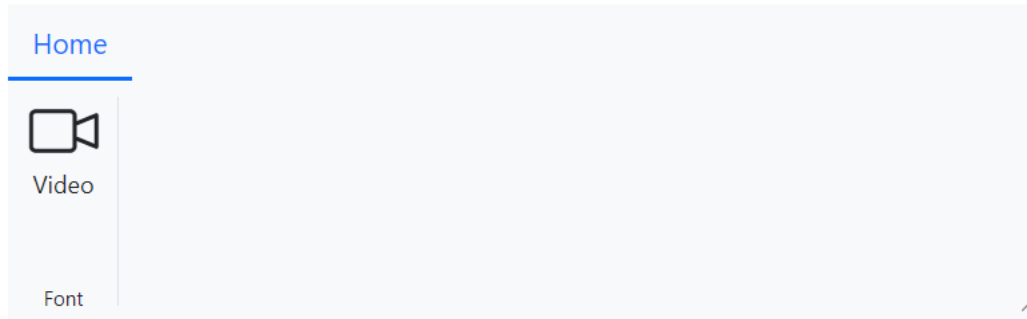


### Custom items

You can customize the ribbon items with non-built-in items or HTML content by setting the [Type](#) property to **Template**. This provides an option to customize the ribbon items with greater flexibility.

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("View").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Template).ItemTemplate("<span class='ribbonTemplate
${activeSize}'>Video").Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<style>
.ribbonTemplate {
display: flex;
align-items: center;
justify-content: center;
cursor: pointer;
}
.ribbonTemplate.Large {
flex-direction: column;
}
.ribbonTemplate.Large .e-icons {
font-size: 35px;
}
.ribbonTemplate.Medium .e-icons,
.ribbonTemplate.Small .e-icons {
font-size: 20px;
margin: 15px 5px;
}
.ribbonTemplate.Small .text {
display: none;
}
</style >
```



### Items display Mode

You can use the [DisplayOptions](#) property to display the items in the Ribbon.

|            |                                                                              |
|------------|------------------------------------------------------------------------------|
| Auto       | The items are displayed in all layouts based on the ribbon's overflow state. |
| Classic    | The items are displayed only in the classic layout group.                    |
| Simplified | The items are displayed only in the simplified layout group.                 |
| Overflow   | The items are displayed only in the overflow popup.                          |

### Display items in Classic only

To display the items only in the classic layout group, set the mode as `DisplayMode.Classic` in the [DisplayOptions](#) property.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).DisplayOptions(Syncfusion.EJ2.Ribbon.DisplayMode.Classic).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
```

### Display items in Simplified only

To display the items only in the simplified layout group, set the mode as `DisplayMode.Simplified` in the [DisplayOptions](#) property.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).DisplayOptions(Syncfusion.EJ2.Ribbon.DisplayMode.Simplified).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render();
 }
```

#### Display items in Overflow popup only

To display the items only in the overflow, set the mode as `DisplayMode.Overflow` in the [DisplayOptions](#) property.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).DisplayOptions(Syncfusion.EJ2.Ribbon.DisplayMode.Overflow).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render();
 }
```

#### Enable or disable items

You can use the [Disabled](#) property to disable a Ribbon item. It prevents the user interaction when set to `true`. By default, the value is `false`.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
```

```

{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).Disabled(true).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 item.Type(RibbonItemType.CheckBox).Disabled(true).CheckBoxSettings(checkBox =>
 {
 checkBox.Checked(true).Label("Ruler");
 }).Add();
 item.Type(RibbonItemType.DropDown).Disabled(true).DropDownSettings(dropdown =>
 {
 dropdown.IconCss("e-icons e-table").Content("Table");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 }
}

```

## Ribbon Layouts

The Ribbon allows to customize the layout by using the [ActiveLayout](#) property. The Ribbon control supports the following layouts:

### Classic layout

In classic layout, the Ribbon control organizes the items and groups in a traditional form by setting the [ActiveLayout](#) property to [Classic](#). By default, the Ribbon control renders in the [Classic](#) layout.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text = "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text = "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").ActiveLayout(RibbonLayout.Classic).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }
 }
)
 }
)
 }
)
}
)

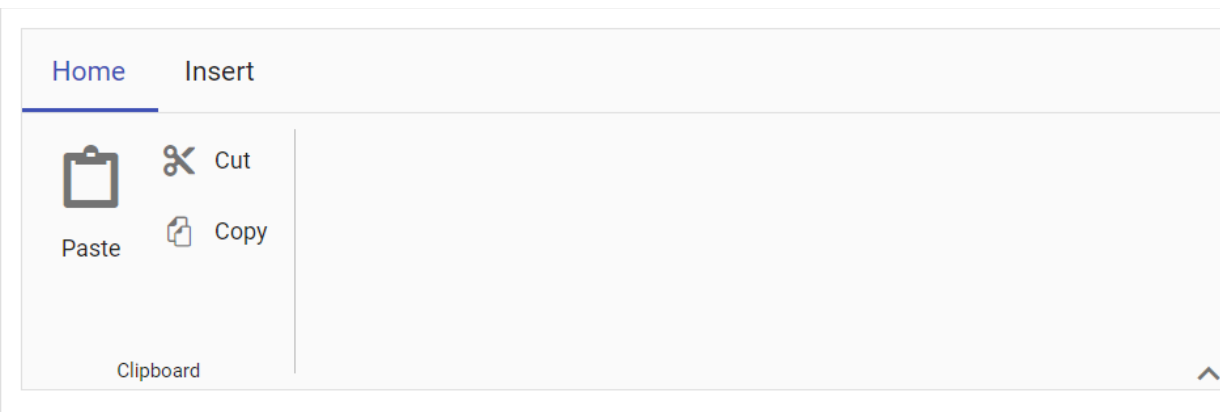
```



```

 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 tab.Header("Insert").Groups(groups =>
 {
 groups.Header("Tables").Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()

```



### Defining items size

You can use the [AllowedSizes](#) property to set the allowed size for an item. The Ribbon items can be appeared in three different sizes: Large(large icon with text), Medium(small icon with text) and Small(small icon only). On resizing, the items size can be changed based on the available width of the tab content from the order of Large-> Medium-> Small and viceversa.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon

```

```

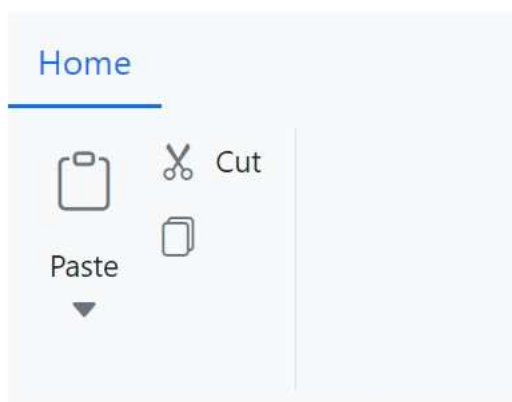
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Collections(collection =>
 {
 collection.Items(item =>
 {

item.Type(RibbonItemType.SplitButton).AllowedSizes(RibbonItemSize.Large).Spl
itButtonSettings(button =>
 {
 button.IconCss("e-icons e-
paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 collection.Items(item =>
 {

item.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Medium).ButtonS
ettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();

item.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Small).ButtonSe
ttings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()

```



*Defining items orientation*

The Ribbon group [Orientation](#) property allows to manage how the items are aligned either in a **Row** or **Column**. By default, the orientation is set to **Column**, in which the items are arranged vertically.

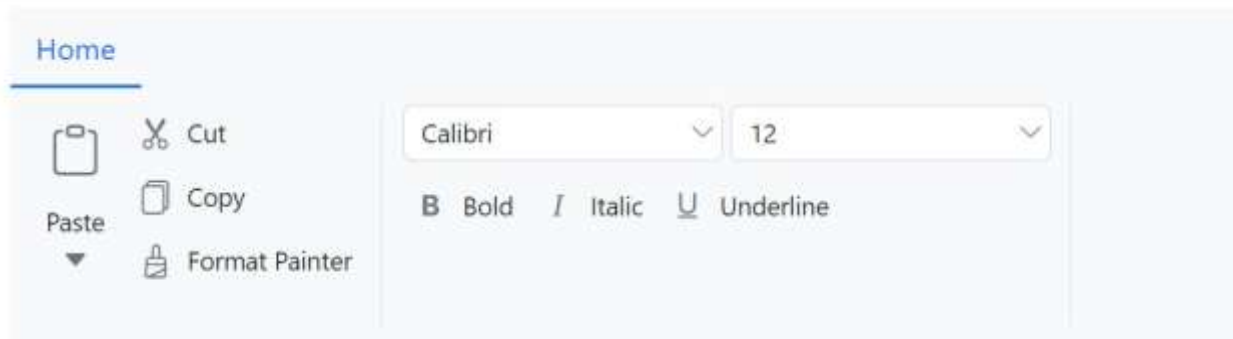
**CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };
 List<string> fontSize = new List<string>() { "8", "9", "10", "11", "12",
"14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72", "96" };
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Orientation(ItemOrientation.Column).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-
paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 group.Orientation(ItemOrientation.Row).Collections(collection =>
 {
 collection.Items(item =>
 {
```

```

=> item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
 {
 button.DataSource(fontStyle).Index(2);
 }).Add();
=> item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
 {
 button.DataSource(fontSize).Index(4);
 }).Add();
collection.Items(item =>
{
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").Content("Bold");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").Content("Italic");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-
underline").Content("Underline");
 }).Add();
 }).Add();
}).Add();
}).Add();
}).Render()

```



When the orientation is set to **Row** a group may have a maximum of three collections each of which may contain any number of items. When the orientation is set to **Column** a group may have any number of collections, each of which may contain one large-sized item or three medium/small-sized items. If two large-sized items are specified, it automatically converts into two medium/small-sized items.

#### [Defining group header](#)

You can use the [Header](#) property to set the name for each group header.

#### **CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-
paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 }).Add();
 group.Header("Font").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").Content("Bold");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").Content("Italic");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-
underline").Content("Underline");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
}

```



### Defining group icon

You can use the [GroupIconCss](#) property to customize the icons in the group overflow button. When the ribbon's size is adjusted, the group popup will appear.

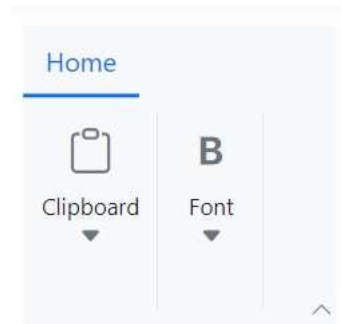
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text = "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem { Text = "Keep text only" } };
 List<string> fontSize = new List<string>() { "8", "9", "10", "11", "12", "14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72", "96" };
 List<string> fontStyle = new List<string>() { "Algerian", "Arial", "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia", "Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").GroupIconCss("e-icons e-paste").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 });
 });
 });
}
```

```

 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 group.Header("Font").GroupIconCss("e-icons e-
bold").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontStyle).Index(2);
 }).Add();
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontSize).Index(4);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").Content("Bold");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").Content("Italic");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-
underline").Content("Underline");
 }).Add();
 }).Add();
 }).Add();
}).Add();
}).Render()

```



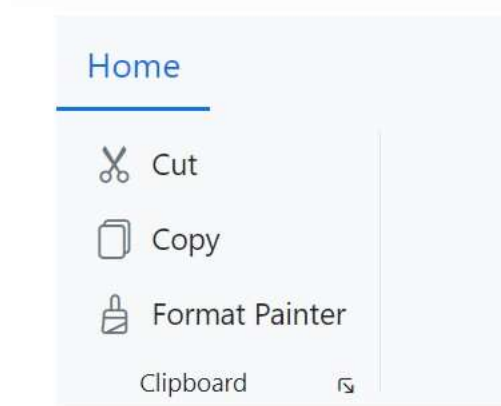
### Enabling group launcher icon

You can use the [ShowLauncherIcon](#) property to enable or disable the launcher icon for each group. By default, the property is set to `false`.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.ShowLauncherIcon(true).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```



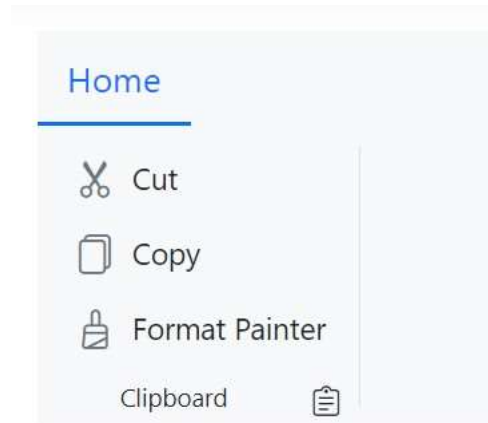


### Customize launcher icon

You can use the [LauncherIconCss](#) property to customize the launcher icon by applying the custom styles.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").LauncherIconCss("e-icons e-
description").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").GroupIconCss("e-icons e-
paste").ShowLauncherIcon(true).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```



### Defining group collapsible state

You can use the [IsCollapsible](#) property to determine whether the group is collapsed or not during resize. By default, the property is set to `true`. To prevent the group from being collapsed, set the property to `false`.

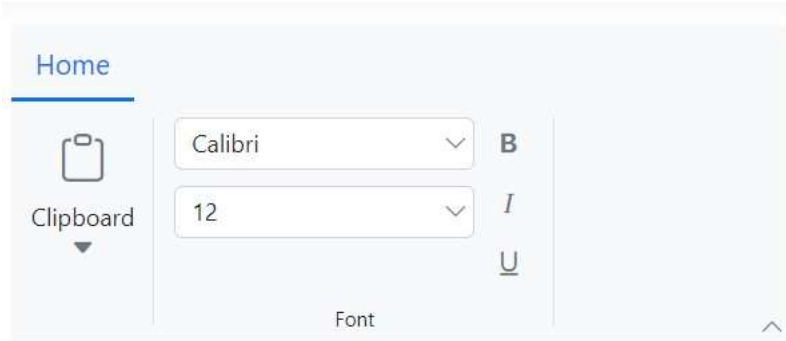
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };
 List<string> fontSize = new List<string>() { "8", "9", "10", "11", "12",
"14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72", "96" };
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").GroupIconCss("e-icons e-
paste").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-
paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
```

```

 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 }).Add();
 group.Header("Font").IsCollapsible(false).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontStyle).Index(2);
 }).Add();
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontSize).Index(4);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").Content("Bold");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").Content("Italic");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-
underline").Content("Underline");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()

```



### Defining priority order for group collapse or expand

You can use the [Priority](#) property to set the priority order for each group which should be collapsed or expanded on resizing. When collapsing, higher priority values are fetched first. When expanding, lower priority values are fetched first.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };
 List<MenuItem> findOptions = new List<MenuItem>() { new MenuItem { Text
= "Find", IconCss = "e-icons e-search" }, new MenuItem { Text = "Advanced
find", IconCss = "e-icons e-search" }, new MenuItem { Text = "Go to",
IconCss = "e-icons e-arrow-right" } };
 List<MenuItem> selectOptions = new List<MenuItem>() { new MenuItem {
Text = "Select All" }, new MenuItem { Text = "Select Objects" } };

 List<string> fontSize = new List<string>() { "8", "9", "10", "11", "12",
"14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72", "96" };
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").GroupIconCss("e-icons e-
paste").Priority(2).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-
paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 }
 }
}
```

```

 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 }).Add();
 group.Header("Font").GroupIconCss("e-icons e-
bold").Priority(0).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontStyle).Index(2);
 }).Add();
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontSize).Index(4);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").Content("Bold");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").Content("Italic");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-
underline").Content("Underline");
 }).Add();
 }).Add();
 }).Add();
 group.Header("Editing").GroupIconCss("e-icons e-
bold").Priority(1).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>

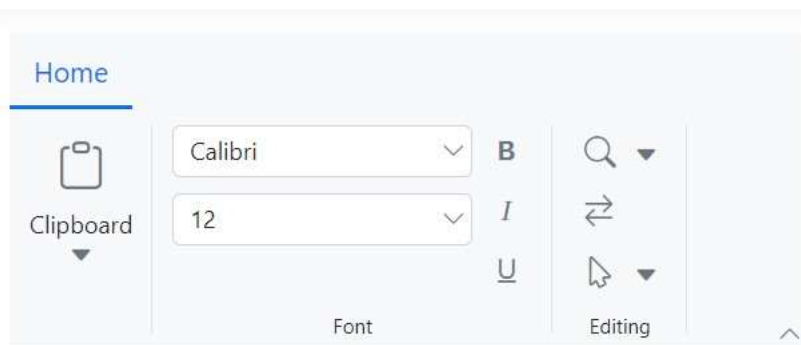
```

```

 {
 button.IconCss("e-icons e-
search").Content("Find").Items(findOptions);
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-replace").Content("Replace");
 }).Add();

 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-mouse-
pointer").Content("Select").Items(selectOptions);
 }).Add();
 }).Add();
}).Add();
}).Add();
}).Render()

```



### Simplified layout

In simplified layout, the Ribbon control organizes the items and groups into a single row by setting the [ActiveLayout](#) property to [Simplified](#).

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text
= "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem {
Text = "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").ActiveLayout(RibbonLayout.Simplified).Tabs(tab
=>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {

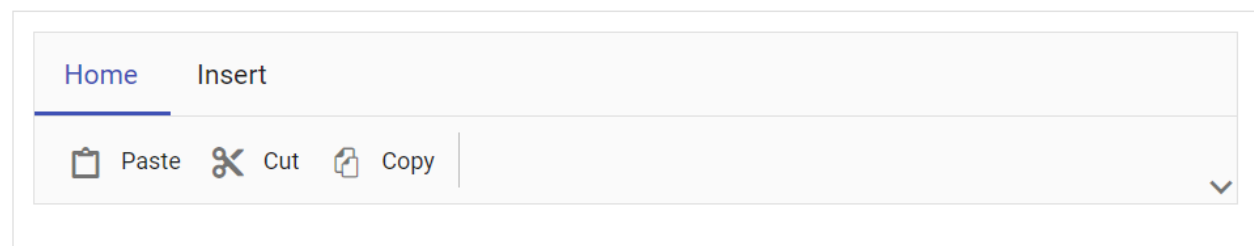
```

```

 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
}).Add();
collection.Items(items =>
{
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
}).Add();
tab.Header("Insert").Groups(groups =>
{
 groups.Header("Tables").Collections(collections =>
 {
 collections.Items(items =>

items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()

```



#### Enabling group overflow popup

You can use the [EnableGroupOverflow](#) property to add a separate popup for the overflow items in the group while resizing. The overflow items will appear in the common overflow popup, located at the right end of the tab content if it is set to **false**.

#### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text
= "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
MenuItem { Text = "Keep text only" } };

```

```

List<MenuItem> findOptions = new List<MenuItem>() { new MenuItem { Text
= "Find", IconCss = "e-icons e-search" }, new MenuItem { Text = "Advanced
find", IconCss = "e-icons e-search" }, new MenuItem { Text = "Go to",
IconCss = "e-icons e-arrow-right" } };

List<MenuItem> selectOptions = new List<MenuItem>() { new MenuItem {
Text = "Select All" }, new MenuItem { Text = "Select Objects" } };

List<string> fontSize = new List<string>() { "8", "9", "10", "11", "12",
"14", "16", "18", "20", "22", "24", "26", "28", "36", "48", "72", "96" };
List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").GroupIconCss("e-icons e-
paste").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-
paste").Content("Paste").Items(pasteOptions);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-
painter").Content("Format Painter");
 }).Add();
 }).Add();
 }).Add();

 group.Header("Font").Orientation(ItemOrientation.Row).GroupIconCss("e-icons
e-bold").EnableGroupOverflow(true).Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontStyle).Index(2);

```

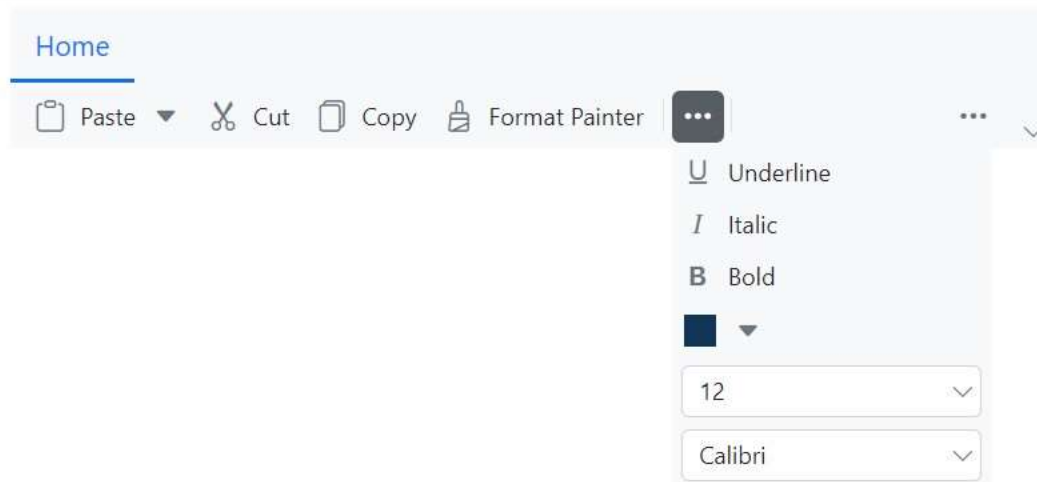


```

 }).Add();
 item.Type(RibbonItemType.ComboBox).ComboBoxSettings(button
=>
 {
 button.DataSource(fontSize).Index(4);
 }).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-bold").Content("Bold");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-italic").Content("Italic");
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-
underline").Content("Underline");
 }).Add();
 }).Add();
 }).Add();
 group.Header("Editing").GroupIconCss("e-icons e-
bold").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-
search").Content("Find").Items(findOptions);
 }).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-replace").Content("Replace");
 }).Add();

 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-mouse-
pointer").Content("Select").Items(selectOptions);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()

```



### Minimized state

You can hide the Ribbon contents and display only the tab headers by double-clicking on the tab header. In minimized state, the Ribbon control expands to its normal state when click on the tab header.

You can use the [IsMinimized](#) property to change the Ribbon component to minimized state. By default, the value is `false`.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text = "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text = "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").IsMinimized(true).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
```

```

 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
}).Add();
tab.Header("Insert").Groups(groups =>
{
 groups.Header("Tables").Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions);
 }).Add();
 }).Add();
 }).Add();
}).Add();
}).Add();
}).Render()

```



## File Menu

The Ribbon control provides a built-in file menu that allows you to add menu items for performing specific actions. The file menu can be enabled by setting the [FileMenu](#) property.

### Visibility

You can show the file menu by setting the [Visible](#) property to `true`. By default, the file menu is hidden.

### CSHTML

```

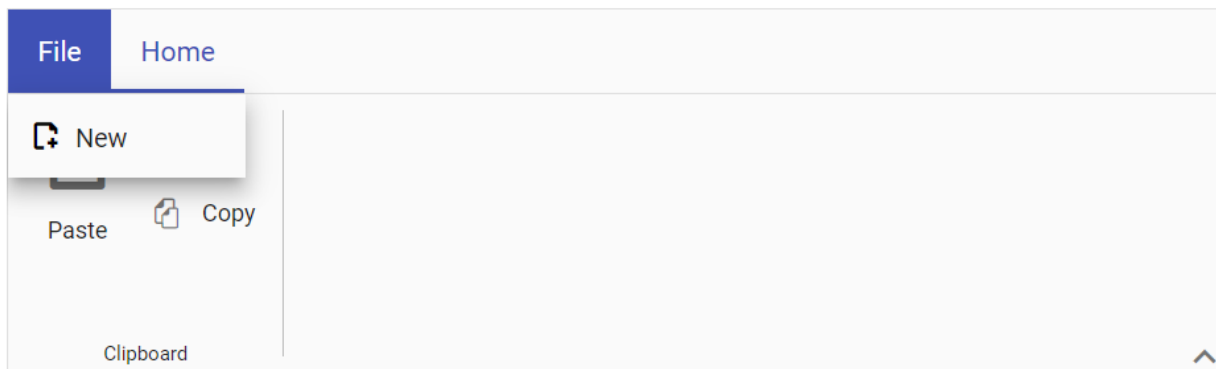
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> fileOptions = new List<MenuItem>()
 {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" }
 };
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Visible(true).MenuItems(fileOptions);
}).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>

```

```

 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
}).Add();
}).Render()

```



### Adding Menu Items

The menu items can be added to the file menu using the [MenuItems](#) property.

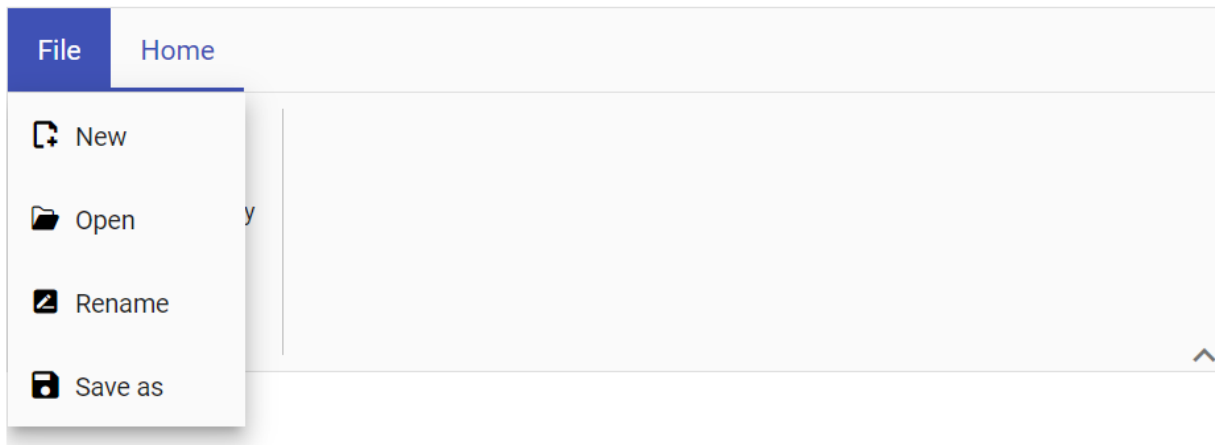
#### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> fileOptions = new List<MenuItem>()
 {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open",
Id="open" },
 new MenuItem { Text = "Rename", IconCss = "e-icons e-rename",
Id="rename" },
 new MenuItem { Text = "Save as", IconCss = "e-icons e-save",
Id="save" }
 };
}

```

```
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Visible(true).MenuItems(fileOptions);
}).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
```



### Open Submenu on click

You can open the submenu on menu item click, by setting the [ShowItemOnClick](#) property to `true`. By default, the submenu will open on mouse hover.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
```

```

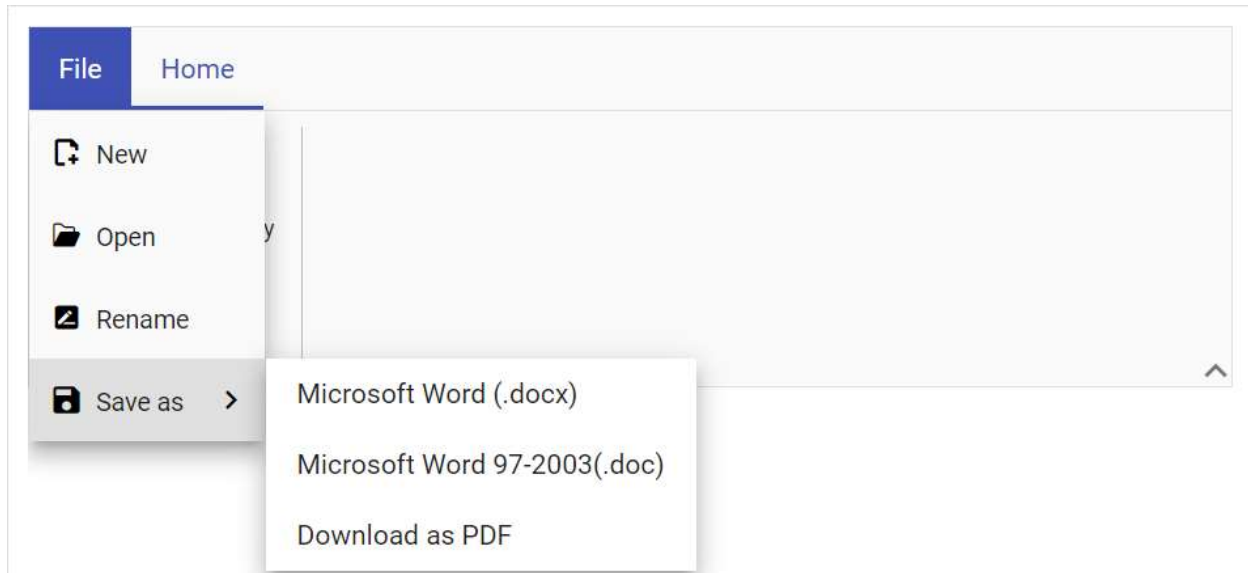
@{
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text
= "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem {
Text = "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
 List<MenuItem> fileOptions = new List<MenuItem>() {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open" },
 new MenuItem { Text = "Rename", IconCss = "e-icons e-rename" },
 new MenuItem { Text = "Save as", IconCss = "e-icons e-save",
 Items= new List<MenuItem>() {
 new MenuItem { Text = "Microsoft Word (.docx)" },
 new MenuItem { Text = "Microsoft Word 97-2003(.doc)" },
 new MenuItem { Text = "Download as PDF" }
 }
 }
 };
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Visible(true).ShowItemOnClick(true).MenuItems(fileOptions);
}).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 tab.Header("Insert").Groups(groups =>
 {
 groups.Header("Tables").Collections(collections =>
 {
 collections.Items(items =>
 {
 items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions);

```

```

 }).Add();
 }).Add();
}).Add();
}).Add();
}).Render();

```



### Custom Header text

You can define the file menu header text content by using the [Text](#) property.

### CSHTML

```

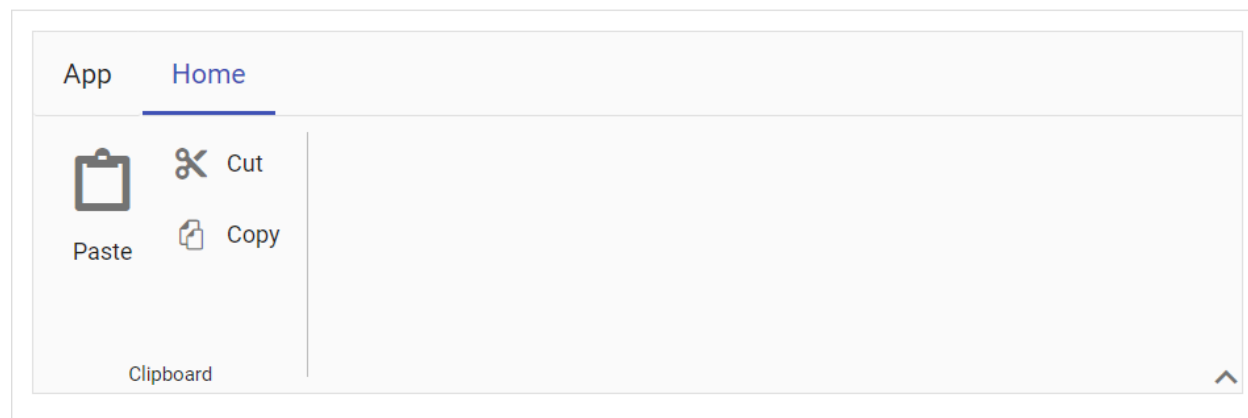
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> fileOptions = new List<MenuItem>()
 {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" }
 };
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Text("App").Visible(true).MenuItems(fileOptions);
}).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}

```

```

collection.Items(items =>
{
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
}).Add();
}).Render()

```



## Ribbon Backstage

The Ribbon supports backstage view which is an addition to the traditional file menu. It displays information like application settings, user details, etc. The backstage view can be enabled by setting the [backStageMenu](#) property.

The backstage view options are displayed on the left, while the content of each option is shown on the right.

## Adding backstage items

The menu items can be added to the backstage view by using the [items](#) property. You can show the backstage view by setting the [visible](#) property to `true`. By default, the backstage view is hidden.

## CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<BackstageItem> backstageItems = new List<BackstageItem>() {
 new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home", Content = processBackstageContent("home") },
 new BackstageItem { Id = "new", Text = "New", IconCss = "e-icons e-file-new", Content = processBackstageContent("new") },
 new BackstageItem { Id = "open", Text = "Open", IconCss = "e-icons e-folder-open", Content = processBackstageContent("open") },
 };
}

```



```

BackStageMenu backstageSettings = new BackStageMenu() { Text = "File",
Visible = true, BackButton = new BackstageBackButton { Text = "Close" },
Items = backstageItems };
}
@functions {
 string processBackstageContent(string item)
 {
 string content = "";
 switch (item)
 {
 case "home":
 {
 content = "<div id='home-wrapper' style='padding:
20px;'><div id='new-section' class='new-wrapper'><div class='section-title'>
New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div><div id='block-wrapper'><div class='section-
title'> Recent </div><div class='section-content' style='padding: 12px 0px;
cursor: pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-
open-link'> </td><td>
Ribbon.docx EJ2 >> Components >>
Navigations >> Ribbon >> default
</td></tr></tbody></table></div></div></div>";
 break;
 }
 case "new":
 {
 content = "<div id='new-section' class='new-wrapper'><div
class='section-title'> New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div>";
 break;
 }
 case "open":
 {
 content = "<div id='open-content' style='padding:
20px;'><div class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-open-
link'> </td><td> Open
in Desktop App Use the full
functionality of Ribbon </td></tr></tbody></table></div><div
class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-protect-
sheet'> </td><td>
Protect Document To prevent accidental
changes, this document has been set to open as view-
only.</td></tr></tbody></table></div></div>";
 break;
 }
 }
 return content;
 }
}
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {

```

```

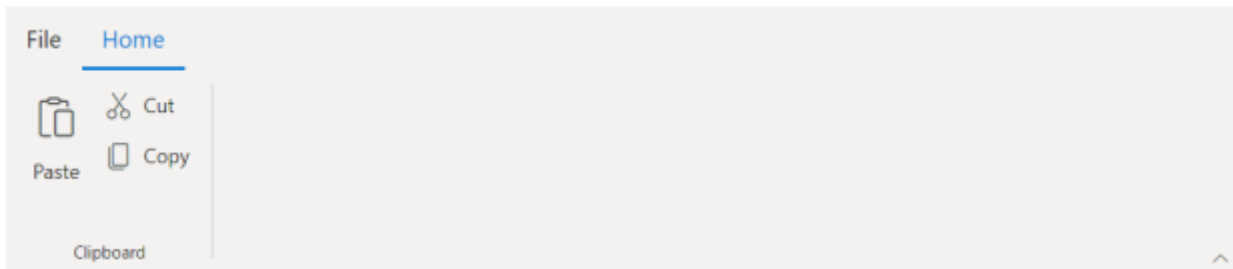
group.Header("Clipboard").Collections(collection =>
{
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
}).Render()
<style>
.e-ribbon-backstage-content{
 width: 500px;
 height: 350px;
}
.section-title {
 font-size: 22px;
}
.new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
}
.category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
}
.doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
}
.doc_category_text {
 font-size: 16px;
}
.section-content {
 padding: 12px 0px;
 cursor: pointer;
}

```

```

 }
 .doc_icon {
 font-size: 16px;
 padding: 0px 10px;
 }
 .category_container:hover, .section-content:hover {
 background-color: #dfdfff;
 border-radius: 5px;
 transition: all 0.3s;
 }
}
</style>

```



### Adding footer items

You can use the [isFooter](#) property in the [items](#) collection to add the backstage view footer items. By default, the value is false.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<BackstageItem> backstageItems = new List<BackstageItem>() {
 new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home", Content = processBackstageContent("home") },
 new BackstageItem { Id = "new", Text = "New", IconCss = "e-icons e-file-new", Content = processBackstageContent("new") },
 new BackstageItem { Id = "open", Text = "Open", IconCss = "e-icons e-folder-open", Content = processBackstageContent("open") },
 new BackstageItem { Separator = true, IsFooter = true },
 new BackstageItem { Text = "Account", IsFooter = true, Content = processBackstageContent("account") }
 };
}

```

```

BackStageMenu backstageSettings = new BackStageMenu() { Text = "File",
Visible = true, BackButton = new BackstageBackButton { Text = "Close" },
Items = backstageItems };
}
@functions {
 string processBackstageContent(string item)
 {
 string content = "";
 switch (item)
 {
 case "home":
 {
 content = "<div id='home-wrapper' style='padding:
20px;'><div id='new-section' class='new-wrapper'><div class='section-title'>
New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div><div id='block-wrapper'><div class='section-
title'> Recent </div><div class='section-content' style='padding: 12px 0px;
cursor: pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-
open-link'> </td><td>
Ribbon.docx EJ2 >> Components >>
Navigations >> Ribbon >> default
</td></tr></tbody></table></div></div></div>";
 break;
 }
 case "new":
 {
 content = "<div id='new-section' class='new-wrapper'><div
class='section-title'> New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div>";
 break;
 }
 case "open":
 {
 content = "<div id='open-content' style='padding:
20px;'><div class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-open-
link'> </td><td> Open
in Desktop App Use the full
functionality of Ribbon </td></tr></tbody></table></div><div
class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-protect-
sheet'> </td><td>
Protect Document To prevent accidental
changes, this document has been set to open as view-
only.</td></tr></tbody></table></div></div>";
 break;
 }
 case "account":
 {
 content = "<div id='account-content' style='padding:
20px;'><div class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-
people'> </td><td> <span style='display: block; font-size:
14px'>Account type<span style='font-size:
12px'>Administrator</td></tr></tbody></table></div><div

```

```

class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-
password'> </td><td> <span style='display: block; font-size:
14px'>Password protected<span style='font-size:
12px'>Yes</td></tr></tbody></table></div></div>";
 break;
 }
}
return content;
}
}
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<style>
.e-ribbon-backstage-content{
 width: 500px;
 height: 350px;
}
.section-title {
 font-size: 22px;
}
.new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
}
.category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
}

```

```

.doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
}
.doc_category_text {
 font-size: 16px;
}
.section-content {
 padding: 12px 0px;
 cursor: pointer;
}
.doc_icon {
 font-size: 16px;
 padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
 background-color: #dfdfff;
 border-radius: 5px;
 transition: all 0.3s;
}
</style>

```



### Adding separator

The separators are horizontal lines used to separate the backstage view items. You can use the [separator](#) property to split the menu items.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon

```

```

@using Syncfusion.EJ2.Navigations
@{
 List<BackstageItem> backstageItems = new List<BackstageItem>() {
 new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home", Content = processBackstageContent("home") },
 new BackstageItem { Id = "new", Text = "New", IconCss = "e-icons e-file-new", Content = processBackstageContent("new") },
 new BackstageItem { Id = "open", Text = "Open", IconCss = "e-icons e-folder-open", Content = processBackstageContent("open") },
 new BackstageItem { Separator = true },
 new BackstageItem { Text = "Print", Content = processBackstageContent("print") }
 };
 BackStageMenu backstageSettings = new BackStageMenu() { Text = "File", Visible = true, BackButton = new BackstageBackButton { Text = "Close" }, Items = backstageItems };
}
@functions {
 string processBackstageContent(string item)
 {
 string content = "";
 switch (item)
 {
 case "home":
 {
 content = "<div id='home-wrapper' style='padding: 20px;'><div id='new-section' class='new-wrapper'><div class='section-title'> New </div><div class='category_container'><div class='doc_category_image'></div> New document </div></div><div id='block-wrapper'><div class='section-title'> Recent </div><div class='section-content' style='padding: 12px 0px; cursor: pointer'><table><tbody><tr><td> </td><td> Ribbon.docx EJ2 >> Components >> Navigations >> Ribbon >> default </td></tr></tbody></table></div></div></div>";
 break;
 }
 case "new":
 {
 content = "<div id='new-section' class='new-wrapper'><div class='section-title'> New </div><div class='category_container'><div class='doc_category_image'></div> New document </div></div>";
 break;
 }
 case "open":
 {
 content = "<div id='open-content' style='padding: 20px;'><div class='section-content' style='padding: 12px 0px; cursor: pointer'><table><tbody><tr><td> </td><td> Open in Desktop App Use the full functionality of Ribbon </td></tr></tbody></table></div><div class='section-content' style='padding: 12px 0px; cursor: pointer'><table><tbody><tr><td> </td><td>

```

```

Protect Document To prevent accidental
changes, this document has been set to open as view-
only.</td></tr></tbody></table></div></div>";
 break;
 }
 case "print":
 {
 content = "<div style='min-width: 300px; padding:
20px;'><h2>Print this document</h2><button id='togglebtn' class='e-control
e-btn e-lib e-flat e-primary'><span class='e-btn-icon e-btn-sb-icons e-icons
e-print e-icon-left'>Print</button></div>";
 break;
 }
 }
 return content;
}
}
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<style>
.e-ribbon-backstage-content{
 width: 500px;
 height: 350px;
}
.section-title {
 font-size: 22px;
}
.new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
}

```



```
.category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
}
.doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
}
.doc_category_text {
 font-size: 16px;
}
.section-content {
 padding: 12px 0px;
 cursor: pointer;
}
.doc_icon {
 font-size: 16px;
 padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
 background-color: #dfdfdf;
 border-radius: 5px;
 transition: all 0.3s;
}
</style>
```



## Back button

You can use the [backButton](#) property to customize the text and icon of the close button using the [text](#) and [iconCss](#) property. You can show the back button by setting the [visible](#) property to `true`.

## CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<BackstageItem> backstageItems = new List<BackstageItem>() {
 new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home", Content = processBackstageContent("home") }
 };
 BackStageMenu backstageSettings = new BackStageMenu() { Text = "File",
 Visible = true, BackButton = new BackstageBackButton { Text = "Close" },
 Items = backstageItems };
}
@functions {
 string processBackstageContent(string item)
 {
 string content = "";
 switch (item)
 {
 case "home":
 {
 content = "<div id='home-wrapper' style='padding: 20px;'><div id='new-section' class='new-wrapper'><div class='section-title'>New</div><div class='category_container'><div class='doc_category_image'></div> New document</div></div><div id='block-wrapper'><div class='section-title'> Recent</div><div class='section-content' style='padding: 12px 0px; cursor: pointer'><table><tbody><tr><td> </td><td> Ribbon.docx EJ2 >> Components >> Navigations >> Ribbon >> default</td></tr></tbody></table></div></div></div>";
 break;
 }
 }
 return content;
 }
}
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
```

```

 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
}).Render()
<style>
.e-ribbon-backstage-content{
 width: 500px;
 height: 350px;
}
.section-title {
 font-size: 22px;
}
.new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
}
.category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
}
.doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
}
.doc_category_text {
 font-size: 16px;
}
.section-content {
 padding: 12px 0px;
 cursor: pointer;
}
.doc_icon {
 font-size: 16px;
 padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
 background-color: #dfdfff;
 border-radius: 5px;
 transition: all 0.3s;
}

```

```
</style>
```



### Backstage target

The [target](#) property specifies the element selector in which backstage will be displayed. The target element should have the position as relative, else the backstage will be positioned nearest to the relative element. By default, the backstage is positioned to ribbon element.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<BackstageItem> backstageItems = new List<BackstageItem>() {
 new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home", Content = processBackstageContent("home") },
 new BackstageItem { Id = "new", Text = "New", IconCss = "e-icons e-file-new", Content = processBackstageContent("new") },
 new BackstageItem { Id = "open", Text = "Open", IconCss = "e-icons e-folder-open", Content = processBackstageContent("open") },
 };
 BackStageMenu backstageSettings = new BackStageMenu() { Text = "File", Visible = true, Target = "#targetElement" BackButton = new BackstageBackButton { Text = "Close" }, Items = backstageItems };
}
@functions {
 string processBackstageContent(string item)
 {
 string content = "";
 switch (item)
 {
 case "home":
 {
 content = "<div id='home-wrapper' style='padding: 20px;'><div id='new-section' class='new-wrapper'><div class='section-title'>
```

```

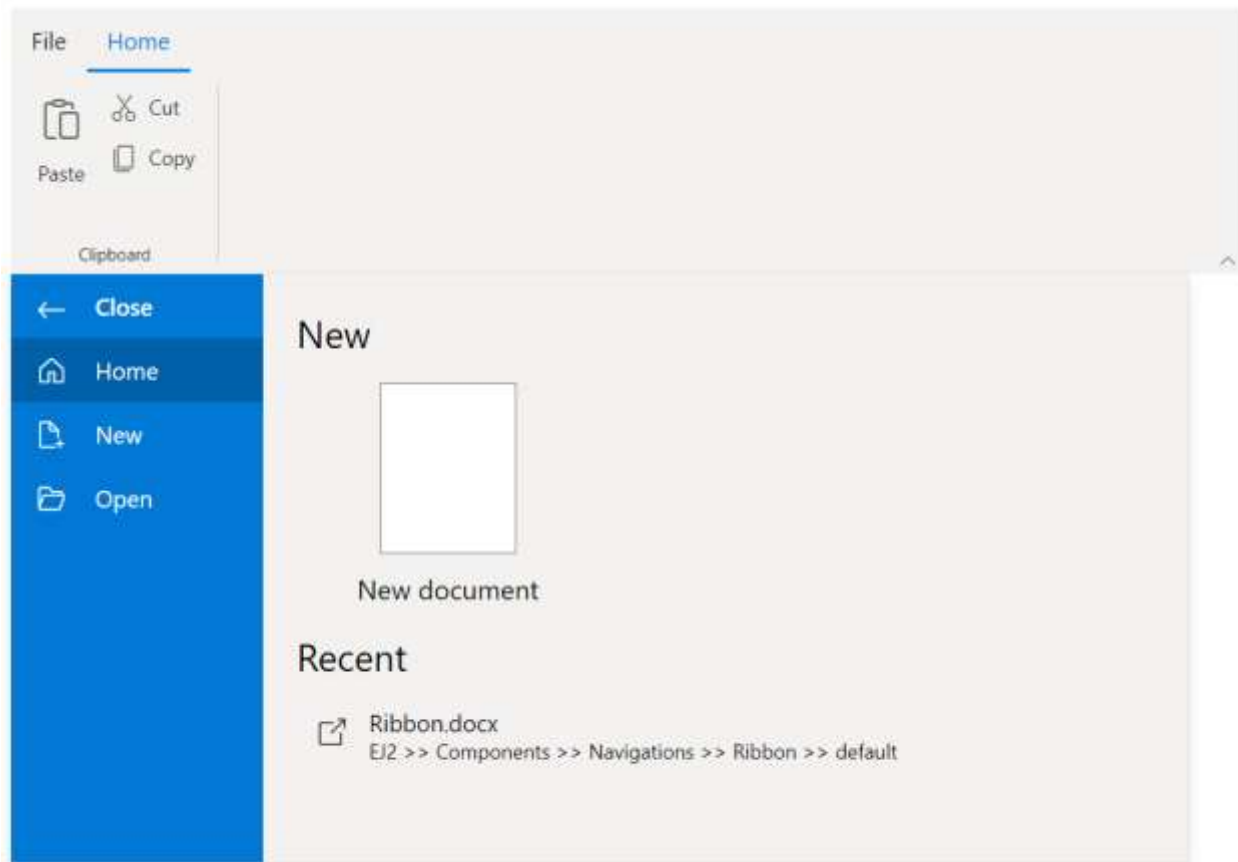
New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div><div id='block-wrapper'><div class='section-
title'> Recent </div><div class='section-content' style='padding: 12px 0px;
cursor: pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-
open-link'> </td><td>
Ribbon.docx EJ2 >> Components >>
Navigations >> Ribbon >> default
</td></tr></tbody></table></div></div></div>";
 break;
 }
 case "new":
 {
 content = "<div id='new-section' class='new-wrapper'><div
class='section-title'> New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div>";
 break;
 }
 case "open":
 {
 content = "<div id='open-content' style='padding:
20px;'><div class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-open-
link'> </td><td> Open
in Desktop App Use the full
functionality of Ribbon </td></tr></tbody></table></div><div
class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-protect-
sheet'> </td><td>
Protect Document To prevent accidental
changes, this document has been set to open as view-
only.</td></tr></tbody></table></div></div>";
 break;
 }
}
return content;
}
}
<div id="targetElement"></div>
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {

```

```

 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<style>
.e-ribbon-backstage-content{
 width: 500px;
 height: 350px;
}
.section-title {
 font-size: 22px;
}
.new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
}
.category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
}
.doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
}
.doc_category_text {
 font-size: 16px;
}
.section-content {
 padding: 12px 0px;
 cursor: pointer;
}
.doc_icon {
 font-size: 16px;
 padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
 background-color: #dfdfdf;
 border-radius: 5px;
 transition: all 0.3s;
}
</style>

```



### Template

You can use the [template](#) property to modify the backstage view menu items and their contents.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 BackStageMenu backstageSettings = new BackStageMenu() { Text = "File",
 Visible = true, BackButton = new BackstageBackButton { Text = "Close" },
 Template = "#templateContent" };
}
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Created("ribbonCreated").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}
```

```

 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script type="text/x-jsrender" id="templateContent">
 <div id="temp-content" style="width: 550px; height: 350px; display:
flex">
 <div id="items-wrapper" style="width: 130px; height:100%;
background: #779de8;">

 <li id="close" onclick="closeContent(this.id)">

 Close

 <li id="new" onclick="contentClick(this.id)">

 New

 <li id="open" onclick="contentClick(this.id)">

 Open

 <li id="save" onclick="contentClick(this.id)">

 Save

 </div>
 <div id="content-wrapper">
 <div id='new-wrapper' class='content-open' style="padding:
20px;">
 <div id='new-section' class='new-wrapper'>
 <div class='section-title'> New </div>
 <div class='category_container'>
 <div class='doc_category_image'></div>
 New document

 </div>
 </div>
 </div>
 <div id="save-wrapper" class='content-close' style="padding:
20px;">
 <div class="section-content" style="padding: 12px 0px;
cursor: pointer">
 <table>
 <tbody>
 <tr>

```



```

 <td> <span class="doc_icon e-icons e-
save"> </td>
 <td>
 <span style="display: block; font-size:
14px"> Save as
 Save as
copy online
 </td>
 </tr>
 </tbody>
 </table>
 </div>
 <div class="section-content" style="padding: 12px 0px
cursor: pointer">
 <table>
 <tbody>
 <tr>
 <td> <span class="doc_icon e-icons e-
rename"> </td>
 <td>
 <span style="display: block; font-size:
14px"> Rename
 Rename
this file.
 </td>
 </tr>
 </tbody>
 </table>
 </div>
 </div>
 <div id="open-wrapper" class='content-close' style="padding:
20px;">
 <div class="section-content" style="padding: 12px 0px;
cursor: pointer">
 <table>
 <tbody>
 <tr>
 <td> <span class="doc_icon e-icons e-open-
link"> </td>
 <td>
 <span style="display: block; font-size:
14px"> Ribbon.docx
 EJ2 >>
Components >> Navigations >> Ribbon >> default
 </td>
 </tr>
 </tbody>
 </table>
 </div>
 <div class="section-content" style="padding: 12px 0px;
cursor: pointer">
 <table>
 <tbody>
 <tr>
 <td> <span class="doc_icon e-icons e-open-
link"> </td>
 <td>

```

```

 <span style="display: block; font-size:
14px"> Classic_layout.docx
 EJ2 >>
Components >> Navigations >> Ribbon >> layouts
 </td>
 </tr>
 </tbody>
 </table>
 </div>
 <div class="section-content" style="padding: 12px 0px;
cursor: pointer">
 <table>
 <tbody>
 <tr>
 <td> <span class="doc_icon e-icons e-open-
link"> </td>
 <td>
 <span style="display: block; font-size:
14px"> Simplified_layout.docx
 EJ2 >>
Components >> Navigations >> Ribbon >> layouts
 </td>
 </tr>
 </tbody>
 </table>
 </div>
 </div>
 </div>
 </div>
 </script>
 <script>
 var ribbon;
 function contentClick(id) {
 let ribbonEle = document.getElementById('ribbon');
 let content = ribbonEle.querySelector('.content-open')
 if(content) { content.classList.replace('content-open', 'content-
close'); }
 ribbonEle.querySelector('#' + id + '-
wrapper').classList.add('content-open');
 }
 function closeContent() {
 let ribbonEle = document.getElementById('ribbon');
 ribbonEle.querySelector('#ribbon_backstagepopup').style.display =
'none'
 }
 function ribbonCreated() {
 ribbon = this;
 ribbon.element.querySelector('.e-ribbon-
backstage').addEventListener('click', function() {
 // Show the #ribbon_backstagepopup element
 ribbon.querySelector('#ribbon_backstagepopup').style.display =
'block';
 });
 }
 </script>
 <style>
 .e-ribbon-backstage-content{

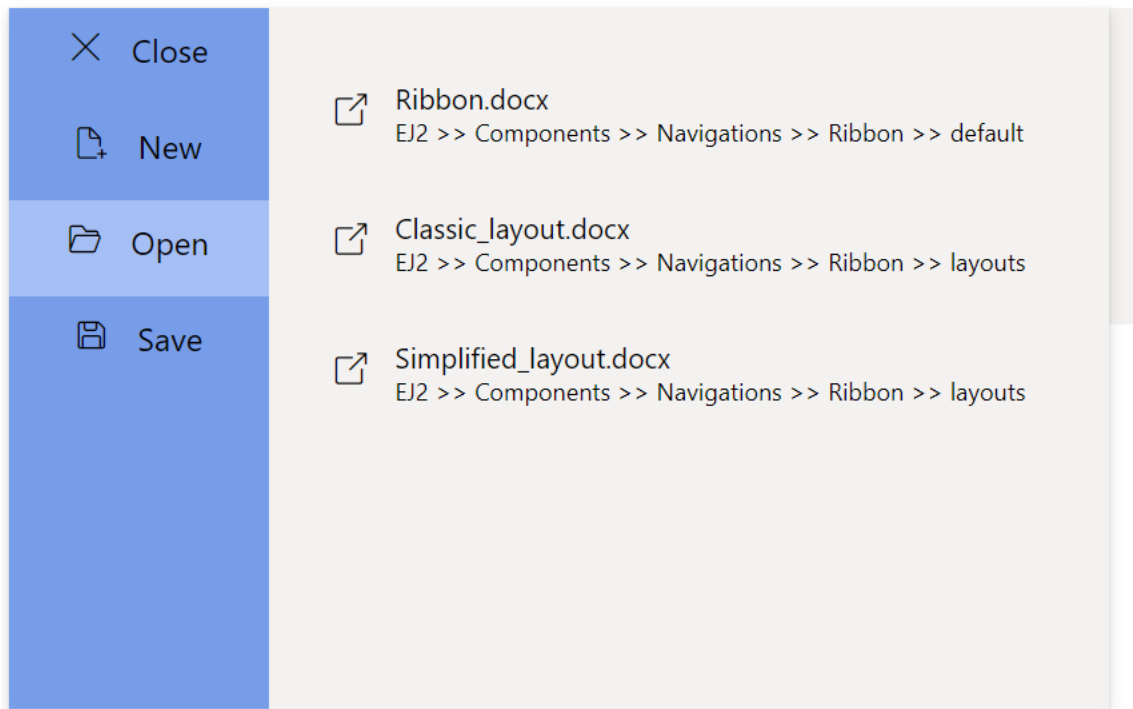
```

```
 width: 500px;
 height: 350px;
 }
 .section-title {
 font-size: 22px;
 }
 .new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
 }
 .category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
 }
 .doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
 }
 .doc_category_text {
 font-size: 16px;
 }
 .section-content {
 padding: 12px 0px;
 cursor: pointer;
 }
 .doc_icon {
 font-size: 16px;
 padding: 0px 10px;
 }
 .category_container:hover, .section-content:hover {
 background-color: #dfdfff;
 border-radius: 5px;
 transition: all 0.3s;
 }
 #targetElement{
 width: 500px;
 height: 500px;
 }
 #items-wrapper ul {
 padding: 0;
 margin: 0;
 }
 #items-wrapper li {
 height: 38px;
 font-size: 16px;
 list-style: none;
 cursor: pointer;
 text-align: center;
 padding-top: 10px;
```

```

 }
 #items-wrapper li span {
 margin-right: 15px;
 font-size: 14px;
 }
 #items-wrapper ul li:hover{
 background-color: #a5bfff;
 }
 #content-wrapper .content-close{
 display: none;
 }
 #content-wrapper .content-open{
 display: block;
 }
}
</style>

```



### Setting width and height

You can customize the height and width of the backstage view using the [height](#) and [width](#) property. By default, dimensions are set based on the content added.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<BackstageItem> backstageItems = new List<BackstageItem>() {
 new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home", Content = processBackstageContent("home") },
 new BackstageItem { Id = "new", Text = "New", IconCss = "e-icons e-file-new", Content = processBackstageContent("new") },
 };
}

```

```

 new BackstageItem { Id = "open", Text = "Open", IconCss = "e-icons
e-folder-open", Content = processBackstageContent("open") },
 };
 BackStageMenu backstageSettings = new BackStageMenu() { Height = "350px"
, Width = "500px" , Text = "File", Visible = true, BackButton = new
BackstageBackButton { Text = "Close" }, Items = backstageItems };
}
@functions {
 string processBackstageContent(string item)
 {
 string content = "";
 switch (item)
 {
 case "home":
 {
 content = "<div id='home-wrapper' style='padding:
20px;'><div id='new-section' class='new-wrapper'><div class='section-title'>
New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div><div id='block-wrapper'><div class='section-
title'> Recent </div><div class='section-content' style='padding: 12px 0px;
cursor: pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-
open-link'> </td><td>
Ribbon.docx EJ2 >> Components >>
Navigations >> Ribbon >> default
</td></tr></tbody></table></div></div></div>";
 break;
 }
 case "new":
 {
 content = "<div id='new-section' class='new-wrapper'><div
class='section-title'> New </div><div class='category_container'><div
class='doc_category_image'></div> New
document </div></div>";
 break;
 }
 case "open":
 {
 content = "<div id='open-content' style='padding:
20px;'><div class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-open-
link'> </td><td> Open
in Desktop App Use the full
functionality of Ribbon </td></tr></tbody></table></div><div
class='section-content' style='padding: 12px 0px; cursor:
pointer'><table><tbody><tr><td> <span class='doc_icon e-icons e-protect-
sheet'> </td><td>
Protect Document To prevent accidental
changes, this document has been set to open as view-
only.</td></tr></tbody></table></div></div>";
 break;
 }
 }
 return content;
 }
}
@Html.EJS().Ribbon("ribbon").BackStageMenu(backstageSettings).Tabs(tab =>

```

```

{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<style>
 .section-title {
 font-size: 22px;
 }
 .new-docs {
 display: flex;
 justify-content: space-around;
 flex-wrap: wrap;
 }
 .category_container {
 width: 150px;
 padding: 15px;
 text-align: center;
 cursor: pointer;
 }
 .doc_category_image {
 width: 80px;
 height: 100px;
 background-color: #fff;
 border: 1px solid rgb(125, 124, 124);
 text-align: center;
 overflow: hidden;
 margin: 0px auto 10px;
 }
 .doc_category_text {
 font-size: 16px;
 }
 .section-content {
 padding: 12px 0px;
 cursor: pointer;
 }
}

```

```

.doc_icon {
 font-size: 16px;
 padding: 0px 10px;
}
.category_container:hover, .section-content:hover {
 background-color: #dfdfdf;
 border-radius: 5px;
 transition: all 0.3s;
}
</style>

```



### [Adding Backstage events](#)

### Help pane

The help pane is dedicated area where the users can define help contents like controlling document permissions, sharing features, and more which appears on the right side of the Ribbon. You can use the [HelpPaneTemplate](#) property to set the help pane contents.

### **INDEX.CSHTML**

```

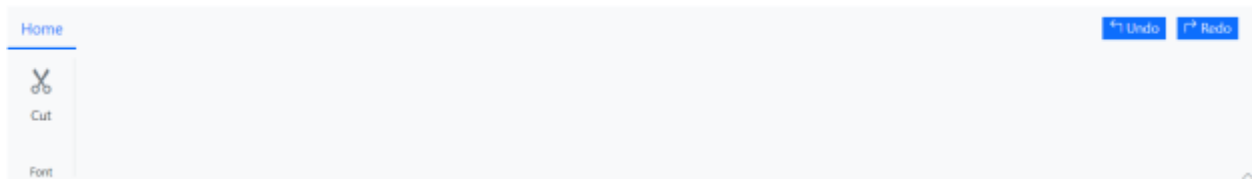
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Ribbon("ribbon").HelpPaneTemplate("#helpPaneTemplate").Tabs(tab
=>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }
 }
 }
}

```

```

 }).Add();
 }).Add();
 }).Add();
 }).Render()
<style>
.action_btn {
margin: 0px 3px;
border: none;
color: #ffffff;
background-color: #0d6efd;
}
#undo, #redo{
padding: 0px 3px ;
}
</style>
<script type="text/x-jsrender" id="helpPaneTemplate">
<button class="action_btn"> <label>
 Undo </label></button>
<button class="action_btn"> <label>
 Redo </label></button>
</script>

```



## Tooltip

The Ribbon component supports tooltip to show additional information in the Ribbon items. The tooltip appears when the user hovers over a Ribbon item.

### Adding Title

You can use the [Title](#) property to set the tooltip title for each Ribbon item.

## CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteMenuOptions = new List<MenuItem>() { new MenuItem { Text
 = "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new
 MenuItem { Text = "Keep text only" } };
 var cutOptions = new RibbonTooltipSettings { Title = "Cut" };
 var copyOptions = new RibbonTooltipSettings { Title = "Copy" };
 var pasteOptions = new RibbonTooltipSettings { Title = "Paste" };
 var formatOptions = new RibbonTooltipSettings { Title = "Format Painter" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>

```



```

{
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Items(pasteMenuOptions).Content("Paste");
 }).RibbonTooltipSettings(pasteOptions).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).RibbonTooltipSettings(cutOptions).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).RibbonTooltipSettings(copyOptions).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-painter").Content("Format Painter");
 }).RibbonTooltipSettings(formatOptions).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
}

```

### Adding Content

You can use the [Content](#) property to set the tooltip content for each Ribbon item.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteMenuOptions = new List<MenuItem>() { new MenuItem { Text = "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem { Text = "Keep text only" } };
 var cutOptions = new RibbonTooltipSettings { Title = "Cut", Content = "Places the selected text or object on the clipboard so that you can paste it somewhere else." };
 var copyOptions = new RibbonTooltipSettings { Title = "Copy", Content = "Copies the chosen text or object to the clipboard so that you can reuse it elsewhere." };
 var pasteOptions = new RibbonTooltipSettings { Title = "Paste", Content = "Insert the clipboard content where the cursor is currently placed." };
 var formatOptions = new RibbonTooltipSettings { Title = "Format Painter", Content = "Copies the formatting style of a selected text or object and applies it to other content within the document." };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>

```

```

{
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Items(pasteMenuOptions).Content("Paste");
 }).RibbonTooltipSettings(pasteOptions).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).RibbonTooltipSettings(cutOptions).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).RibbonTooltipSettings(copyOptions).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-painter").Content("Format Painter");
 }).RibbonTooltipSettings(formatOptions).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
}

```

### Adding Icon

You can use the [IconCss](#) property to specify the icons to be displayed in the tooltip.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteMenuOptions = new List<MenuItem>() { new MenuItem { Text = "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem { Text = "Keep text only" } };
 var cutOptions = new RibbonTooltipSettings { Title = "Cut", Content = "Places the selected text or object on the clipboard so that you can paste it somewhere else.", IconCss = "e-icons e-cut" };
 var copyOptions = new RibbonTooltipSettings { Title = "Copy", Content = "Copies the chosen text or object to the clipboard so that you can reuse it elsewhere.", IconCss = "e-icons e-copy" };
 var pasteOptions = new RibbonTooltipSettings { Title = "Paste", Content = "Insert the clipboard content where the cursor is currently placed.", IconCss = "e-icons e-paste" };
 var formatOptions = new RibbonTooltipSettings { Title = "Format Painter", Content = "Copies the formatting style of a selected text or object and applies it to other content within the document.", IconCss = "e-icons e-format-painter" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>

```

```

{
collection.Items(item =>
{
item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
{
button.IconCss("e-icons e-paste").Items(pasteMenuOptions).Content("Paste");
}).RibbonTooltipSettings(pasteOptions).Add();
}).Add();
collection.Items(item =>
{
item.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).RibbonTooltipSettings(cutOptions).Add();
item.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-copy").Content("Copy");
}).RibbonTooltipSettings(copyOptions).Add();
item.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-format-painter").Content("Format Painter");
}).RibbonTooltipSettings(formatOptions).Add();
}).Add();
}).Add();
}).Add();
}).Render()

```

### Customization

You can use the [CssClass](#) property to customize the appearance of the tooltip with your own custom styles.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> pasteMenuOptions = new List<MenuItem>() { new MenuItem { Text = "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem { Text = "Keep text only" } };
var cutOptions = new RibbonTooltipSettings { Title = "Cut", Content = "Places the selected text or object on the clipboard so that you can paste it somewhere else.", IconCss = "e-icons e-cut", CssClass = "custom-tooltip" };
var copyOptions = new RibbonTooltipSettings { Title = "Copy", Content = "Copies the chosen text or object to the clipboard so that you can reuse it elsewhere.", IconCss = "e-icons e-copy", CssClass = "custom-tooltip" };
var pasteOptions = new RibbonTooltipSettings { Title = "Paste", Content = "Insert the clipboard content where the cursor is currently placed.", IconCss = "e-icons e-paste", CssClass = "custom-tooltip" };
var formatOptions = new RibbonTooltipSettings { Title = "Format Painter", Content = "Copies the formatting style of a selected text or object and applies it to other content within the document.", IconCss = "e-icons e-format-painter", CssClass = "custom-tooltip" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>

```

```

{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.SplitButton).SplitButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Items(pasteMenuOptions).Content("Paste");
 }).RibbonTooltipSettings(pasteOptions).Add();
 }).Add();
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).RibbonTooltipSettings(cutOptions).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).RibbonTooltipSettings(copyOptions).Add();
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-format-painter").Content("Format Painter");
 }).RibbonTooltipSettings(formatOptions).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 }
}
<style>
:root {
--borderColor: rgb(72, 72, 72);
--black: #000000;
}
/* To customize the appearance of the tooltip */
.custom-tooltip.e-ribbon-tooltip.e-popup {
border: 2px solid var(--borderColor);
border-radius: 5px;
background: var(--black);
}
/* To customize the arrow of the tooltip */
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip .e-arrow-tip-inner.e-tip-top,
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip .e-arrow-tip-inner.e-tip-
bottom {
color: var(--black);
}
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip-outer.e-tip-top {
border-bottom: 8px solid var(--borderColor);
}
.custom-tooltip.e-ribbon-tooltip .e-arrow-tip-outer.e-tip-bottom {
border-top: 8px solid var(--borderColor);
}
/* To change the size of the tooltip title */
.custom-tooltip.e-ribbon-tooltip .e-tip-content .e-ribbon-tooltip-title {
font-size: 14px;
}

```

```
/* To change the size of the tooltip content */
.custom-tooltip.e-ribbon-tooltip .e-tip-content .e-ribbon-text-container .e-
ribbon-tooltip-content {
font-size: 11px;
}
</style>
```

## Ribbon Resizing

The Ribbon effectively resizes the ribbon elements while being resized. It extends when the ribbon size is increased and collapses when the ribbon size is decreased. The resizing can be performed in both the classic and simplified modes. Also, we have an option to resize the ribbon elements in the custom order.

In classic mode on resizing, the items size will be changed based on the available width of the tab content from the order of Large-> Medium-> Small and viceversa.

In simplified mode on resizing, the items size will be changed based on the available width of the tab content from the order of Medium-> Small and viceversa.

### Defining items allowed size

You can use the [AllowedSizes](#) property to maintain a constant size for an item. If **AllowedSizes** is set, it keeps the size constant even when being resized.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Collections(collection =>
{
collection.Items(item =>
{
item.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Large).ButtonSe
ttings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
```

### Defining items active size

You can use the [ActiveSize](#) property to define the item size initially, before it is being resized. When resized the **ActiveSize** property is updated based on the ribbon's overflow state, which is determined by the **AllowedSizes** property being configured. By default, the value is **Medium**.

## Events

This section describes the ribbon events that will be triggered when appropriate actions are performed. The following events are available in the ribbon control.

### Tab selected

The [TabSelected](#) event is triggered after selecting the tab item.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").TabSelected("function(args) {tabSelectedEvent(ar
gs)}").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 <script>
 function tabSelectedEvent(args) {
 // Here, you can customize your code.
 }
 </script>
```

### Tab selecting

The [TabSelecting](#) event is triggered before selecting the tab item.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").TabSelecting("function(args) {tabSelectingEvent (
args)}").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
```

```

{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-paste").Content("Paste");
}).Add();
}).Add();
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-copy").Content("Copy");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function tabSelectingEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Ribbon collapsing

The [RibbonCollapsing](#) event is triggered before collapsing the ribbon.

#### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").RibbonCollapsing("function(args){ribbonCollapsingEvent(args)}").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-paste").Content("Paste");
}).Add();
}).Add();
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-copy").Content("Copy");
}

```

```

 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function ribbonCollapsingEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### Ribbon expanding

The [RibbonExpanding](#) event is triggered before expanding the ribbon.

#### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").RibbonExpanding("function(args){ribbonExpanding
(args)}").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function ribbonExpanding(args) {
 // Here, you can customize your code.
}
</script>

```

### Group launcher click

The [LauncherIconClick](#) event is triggered when the launcher icon of the group is clicked.



**INDEX.CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").LauncherIconClick("function(args){launchClick(a
rgs)}").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").ShowLauncherIcon(true).Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-paste").Content("Paste");
 }).Add();
 }).Add();
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("Copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 <script>
 function launchClick(args) {
 // Here, you can customize your code.
 }
 </script>

```

## Button item events

*Clicked*

The [Clicked](#) event is triggered when the button is clicked.

**INDEX.CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {

```

```
button.IconCss("e-icons e-cut").Content("Cut").Clicked("function() { clickedEvent() }");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function clickedEvent() {
// Here, you can customize your code.
}
</script>
```

### Created

The [Created](#) event is triggered when the button is created.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut").Created("function() { createdEvent() }");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function createdEvent() {
// Here, you can customize your code.
}
</script>
```

### Checkbox item events

#### Change

The [Change](#) event is triggered when the Checkbox state is changed.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
```

```

group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
{
checkBox.Label("Ruler").Checked(false).Change("function(){changeEvent()}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function changeEvent() {
// Here, you can customize your code.
}
</script>

```

### Created

The [Created](#) event is triggered once the Checkbox is created.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.CheckBox).CheckBoxSettings(checkBox =>
{
checkBox.Label("Ruler").Checked(false).Created("function(){createdEvent()}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function createdEvent() {
// Here, you can customize your code.
}
</script>

```

### Colorpicker item events

#### Change

The [Change](#) event is triggered while changing the colors.

### INDEX.CSHTML

```

@using Syncfusion.EJ2

```

```
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").Change("function() {changeEvent()}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 <script>
 function changeEvent() {
 // Here, you can customize your code.
 }
 </script>
```

#### Created

The [Created](#) event is triggered once the ColorPicker is created.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").Created("function() {createdEvent()}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 <script>
 function createdEvent() {
 // Here, you can customize your code.
 }
 </script>
```

#### Open

The [Open](#) event is triggered while opening the ColorPicker popup.

**INDEX.CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").Open("function(args) {openEvent(args)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function openEvent(args) {
 // Here, you can customize your code.
}
</script>
```

*Select*

The [Select](#) event is triggered while selecting the color in picker/palette, when showButtons property is enabled.

**INDEX.CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").Select("function(args) {selectEvent(args)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function selectEvent(args) {
 // Here, you can customize your code.
}
</script>
```

*Before close*

The [BeforeClose](#) event is triggered before closing the ColorPicker popup.

**INDEX.CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").BeforeClose("function(args){beforeCloseEvent(ar
gs)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function beforeCloseEvent() {
 // Here, you can customize your code.
}
</script>
```

*Before open*

The [BeforeOpen](#) event is triggered before opening the ColorPicker popup.

**INDEX.CSHTML**

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").BeforeOpen("function(args){beforeOpenEvent(args
)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function beforeOpenEvent(args) {
 // Here, you can customize your code.
}
```

```
}
</script>
```

### Before tile render

The [BeforeTileRender](#) event is triggered while rendering each palette tile.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorPicker =>
 {
 colorPicker.Value("#123456").BeforeTileRender("function(args) {beforeTileRenderEvent(args)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function beforeTileRenderEvent(args) {
 // Here, you can customize your code.
}
</script>
```

### ComboBox item events

#### Change

The [Change](#) event is triggered when an item in a popup is selected or when the model value is changed by the user.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@{
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
 "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
 "Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
 Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
```

```
{
items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
{
comboBox.DataSource(fontStyle).Index(2).Change("function(args){changeEvent(a
rgs)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function changeEvent() {
// Here, you can customize your code.
}
</script>
```

### Close

The [Close](#) event is triggered when the popup is closed.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@{
List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
{
comboBox.DataSource(fontStyle).Index(2).Close("function(args){closeEvent(arg
s)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function closeEvent(args) {
// Here, you can customize your code.
}
</script>
```

### Open

The [Open](#) event is triggered when the popup is opened.



**INDEX.CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@{
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
 "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
 "Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
 Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
 {
 comboBox.DataSource(fontStyle).Index(2).Open("function(args) {openEvent(args)
 }");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 }
 <script>
 function openEvent(args) {
 // Here, you can customize your code.
 }
 </script>

```

*Created*

The [Created](#) event is triggered once the Combobox is created.

**INDEX.CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@{
 List<string> fontStyle = new List<string>() { "Algerian", "Arial",
 "Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
 "Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
 Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
 {

```

```

comboBox.DataSource(fontStyle).Index(2).Created("function(args){createdEvent
(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function createdEvent() {
// Here, you can customize your code.
}
</script>

```

### Filtering

The [Filtering](#) event triggers on typing a character in the Combobox.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@{
List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
{
comboBox.DataSource(fontStyle).Index(2).Filtering("function(args){filteringE
vent(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function filteringEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Select

The [Select](#) event is triggered when an item in the popup is selected.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon

```

```
@{
List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
{
comboBox.DataSource(fontStyle).Index(2).Select("function(args){selectEvent(a
rgs)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function selectEvent(args) {
// Here, you can customize your code.
}
</script>
```

### Before open

The [BeforeOpen](#) event triggers before opening the popup.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@{
List<string> fontStyle = new List<string>() { "Algerian", "Arial",
"Calibri", "Cambria", "Cambria Math", "Courier New", "Candara", "Georgia",
"Impact", "Segoe Print", "Segoe Script", "Segoe UI", "Symbol", "Times New
Roman", "Verdana", "Windings" };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.ComboBox).ComboBoxSettings(comboBox =>
{
comboBox.DataSource(fontStyle).Index(2).BeforeOpen("function(args){beforeOpe
nEvent(args)}");
}).Add();
}).Add();
}
```

```

}).Add();
}).Add();
}).Render()
<script>
function beforeOpenEvent(args) {
// Here, you can customize your code.
}
</script>

```

## DropDown item events

### Before close

The [BeforeClose](#) event is triggered before closing the DropDownButton popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
"Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text =
"Convert Table" }, new MenuItem { Text = "Excel Spreadsheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Tables").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
{
dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions).BeforeClose("function(args){bef
oreCloseEvent(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function beforeCloseEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Before open

The [BeforeOpen](#) event is triggered before opening the DropDown button popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations

```

```
@{
List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
"Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text
= "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Tables").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
{
dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions).BeforeOpen("function(args){befo
reOpenEvent(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function beforeOpenEvent(args) {
// Here, you can customize your code.
}
</script>
```

### Before item render

The [BeforeItemRender](#) event is triggered while rendering each popup item of the Dropdown button.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
"Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text
= "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Tables").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
{
dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions).BeforeItemRender("function(args
){beforeItemRenderEvent(args)}");
}).Add();
```

```

 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function beforeItemRenderEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### Open

The [Open](#) event is triggered while opening the Dropdown button popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
 "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text =
 "Convert Table" }, new MenuItem { Text = "Excel Spreadsheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Tables").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-
 table").Content("Table").Items(tableOptions).Open("function(args) {openEvent (
 args) }");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 <script>
 function openEvent(args) {
 // Here, you can customize your code.
 }
 </script>

```

### Close

The [Close](#) event is triggered while closing the Dropdown button popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{

```

```

List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
"Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text
= "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Tables").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
{
dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions).Close("function(args){closeEven
t(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function closeEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Created

The [Created](#) event is triggered when the DropDown is created.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
"Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text
= "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Tables").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
{
dropDown.IconCss("e-icons e-
table").Content("Table").Items(tableOptions).Created("function(args){created
Event(args)}");
}).Add();
}).Add();
}).Add();
}).Render()

```

```

 }).Add();
 }).Add();
 }).Render()
<script>
function createdEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### Select

The [Select](#) event is triggered while selecting an action item in the Dropdown button popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> tableOptions = new List<MenuItem>() { new MenuItem { Text =
 "Insert Table" }, new MenuItem { Text = "This device" }, new MenuItem { Text
 = "Convert Table" }, new MenuItem { Text = "Excel SpreadSheet" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Tables").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.DropDown).DropDownSettings(dropDown =>
 {
 dropDown.IconCss("e-icons e-
 table").Content("Table").Items(tableOptions).Select("function(args) {selectEv
 ent(args)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function selectEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### SplitButton item events

#### Before close

The [BeforeClose](#) event is triggered before closing the SplitButton popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations

```



```
@{
List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").BeforeClose("function(args){bef
oreCloseEvent(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function beforeCloseEvent(args) {
// Here, you can customize your code.
}
</script>
```

### Before open

The [BeforeOpen](#) event is triggered before opening the SplitButton popup.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").BeforeOpen("function(args){befo
reOpenEvent(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function beforeOpenEvent(args) {
// Here, you can customize your code.
}
</script>
```

```

 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function beforeOpenEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### Before item render

The [BeforeItemRender](#) event is triggered while rendering each popup item of SplitButton.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
 "Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
 { Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
 {
 splitButton.IconCss("e-icons e-
 paste").Items(pasteOptions).Content("Paste").BeforeItemRender("function(args
){beforeItemRenderEvent(args)}");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function beforeItemRenderEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### Open

The [Open](#) event is triggered while opening the SplitButton popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{

```

```

List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").Open("function(args){openEvent(
args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function openEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Close

The [Close](#) event is triggered while closing the SplitButton popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").Close("function(args){closeEven
t(args)}");
}).Add();
}).Add();
}).Add();
}).Render()

```

```

 }).Add();
 }).Add();
 }).Render()
<script>
function closeEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Created

The [Created](#) event is triggered when the SplitButton is created.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").Created("function(args){created
Event(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function createdEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Select

The [Select](#) event is triggered while selecting an action item in the SplitButton popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{

```

```

List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").Select("function(args){selectEv
ent(args)}");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function selectEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Click

The [Click](#) event is triggered while clicking the primary button in the SplitButton.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> pasteOptions = new List<MenuItem>() { new MenuItem { Text =
"Keep Source Format" }, new MenuItem { Text = "Merge format" }, new MenuItem
{ Text = "Keep text only" } };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.SplitButton).SplitButtonSettings(splitButton =>
{
splitButton.IconCss("e-icons e-
paste").Items(pasteOptions).Content("Paste").Click("function(args){clickEven
t(args)}");
}).Add();
}).Add();
}).Add();
}).Render()

```

```

 }).Add();
 }).Add();
 }).Render()
<script>
function clickEvent(args) {
 // Here, you can customize your code.
}
</script>

```

## GroupButton item events

### BeforeClick

The [BeforeClick](#) event is triggered before selecting a button from the groupbutton items.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> events = new List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-bold", Content = "Bold",
 BeforeClick = "function(args){beforClickEvent(args)}" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-italic", Content = "Italic",
 Selected = true, BeforeClick = "function(args){beforClickEvent(args)}" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-underline", Content =
 "Underline", BeforeClick = "function(args){beforClickEvent(args)}" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-strikethrough", Content =
 "Strikethrough", BeforeClick = "function(args){beforClickEvent(args)}" }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).GroupButtonSettings(groupButton =>
 {
 groupButton.Selection(RibbonGroupButtonSelection.Multiple).Items(events);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function beforClickEvent(args) {
 // Here, you can customize your code.
}
</script>

```

### Click

The [Click](#) event is triggered when selecting a button from the groupbutton items.

**INDEX.CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<RibbonGroupButtonItem> events = new List<RibbonGroupButtonItem>() {
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-left", Content =
 "Align Left", Click = "function(args){clickEvent(args)}" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-center", Content =
 "Align Center", Click = "function(args){clickEvent(args)}", Selected = true
 },
 new RibbonGroupButtonItem { IconCss = "e-icons e-align-right", Content =
 "Align Right", Click = "function(args){clickEvent(args)}" },
 new RibbonGroupButtonItem { IconCss = "e-icons e-justify", Content =
 "Justify", Click = "function(args){clickEvent(args)}" }
 };
}
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Paragraph").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.GroupButton).AllowedSizes(RibbonItemSize.Small).GroupButtonSettings(groupButton =>
 {
 groupButton.Selection(RibbonGroupButtonSelection.Single).Items(events);
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 <script>
 function clickEvent(args) {
 // Here, you can customize your code.
 }
 </script>
}

```

## FileMenu events

*Before close*

The [BeforeClose](#) event is triggered before closing the FileMenu popup.

**INDEX.CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> fileOptions = new List<MenuItem>()
 {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open", Id="open"
 },
}

```

```

new MenuItem { Text = "Rename", IconCss = "e-icons e-rename", Id="rename" },
new MenuItem { Text = "Save as", IconCss = "e-icons e-save", Id="save" }
};
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
file.Visible(true).MenuItems(fileOptions).BeforeClose("function(args){before
CloseEvent(args)}");
}).Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function beforeCloseEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Before open

The [BeforeOpen](#) event is triggered before opening the FileMenu popup.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> fileOptions = new List<MenuItem>()
{
new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open", Id="open"
},
new MenuItem { Text = "Rename", IconCss = "e-icons e-rename", Id="rename" },
new MenuItem { Text = "Save as", IconCss = "e-icons e-save", Id="save" }
};
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
file.Visible(true).MenuItems(fileOptions).BeforeOpen("function(args){beforeO
penEvent(args)}");
}).Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{

```



```

group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function beforeOpenEvent(args) {
// Here, you can customize your code.
}
</script>

```

### Before item render

The [BeforeItemRender](#) event is triggered while rendering each ribbon FileMenu item.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> fileOptions = new List<MenuItem>()
{
new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open", Id="open"
},
new MenuItem { Text = "Rename", IconCss = "e-icons e-rename", Id="rename" },
new MenuItem { Text = "Save as", IconCss = "e-icons e-save", Id="save" }
};
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
file.Visible(true).MenuItems(fileOptions).BeforeItemRender("function(args){beforeItemRenderEvent(args)}");
}).Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
}
}

```

```
<script>
function beforeItemRenderEvent(args) {
 // Here, you can customize your code.
}
</script>
```

### Open

The [Open](#) event is triggered when the FileMenu popup is opened.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
 List<MenuItem> fileOptions = new List<MenuItem>()
 {
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open", Id="open"
 },
 new MenuItem { Text = "Rename", IconCss = "e-icons e-rename", Id="rename" },
 new MenuItem { Text = "Save as", IconCss = "e-icons e-save", Id="save" }
 };
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Visible(true).MenuItems(fileOptions).Open("function(args) {openEvent(args)}");
}).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function openEvent(args) {
 // Here, you can customize your code.
}
</script>
```

### Close

The [Close](#) event is triggered when FileMenu popup is closed.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
```

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> fileOptions = new List<MenuItem>()
{
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open", Id="open"
 },
 new MenuItem { Text = "Rename", IconCss = "e-icons e-rename", Id="rename" },
 new MenuItem { Text = "Save as", IconCss = "e-icons e-save", Id="save" }
};
}
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
 file.Visible(true).MenuItems(fileOptions).Close("function(args){closeEvent(a
rgs)}");
}).Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(items =>
 {
 items.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("Cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 }
 <script>
 function closeEvent(args) {
 // Here, you can customize your code.
 }
 </script>

```

### Select

The [Select](#) event is triggered while selecting an item in the ribbon FileMenu.

### INDEX.CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<MenuItem> fileOptions = new List<MenuItem>()
{
 new MenuItem { Text = "New", IconCss = "e-icons e-file-new" },
 new MenuItem { Text = "Open", IconCss = "e-icons e-folder-open", Id="open"
 },
 new MenuItem { Text = "Rename", IconCss = "e-icons e-rename", Id="rename" },
 new MenuItem { Text = "Save as", IconCss = "e-icons e-save", Id="save" }
};
}

```

```
@Html.EJS().Ribbon("ribbon").FileMenu(file =>
{
file.Visible(true).MenuItems(fileOptions).Select("function(args){selectEvent
(args)}");
}).Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function selectEvent(args) {
// Here, you can customize your code.
}
</script>
```

## Backstage Menu events

### [BackStageItemClick](#)

The [BackStageItemClick](#) event is triggered when backstage item is selected.

### INDEX.CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@{
List<BackstageItem> backstageItems = new List<BackstageItem>() {
new BackstageItem { Id = "home", Text = "Home", IconCss = "e-icons e-home",
Content = processBackstageContent("home"), BackStageItemClick =
backStageItemClickEvent(args) },
};
BackStageMenu backstageSettings = new BackStageMenu() { Text = "File",
Visible = true, BackButton = new BackstageBackButton { Text = "Close" },
Items = backstageItems };
@Html.EJS().Ribbon("backstage-
ribbon").BackStageMenu(backstageSettings).Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(items =>
{
items.Type(RibbonItemType.Button).ButtonSettings(button =>
{
```

```
button.IconCss("e-icons e-cut").Content("Cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function backStageItemClickEvent(args) {
// Here, you can customize your code.
}
</script>
```

## Methods

The following methods are available in the Ribbon control.

### addTab

Using the **addTab** method you can add tab dynamically in the ribbon control. The arguments are as follows.

| Argument name       | Description                                                  |
|---------------------|--------------------------------------------------------------|
| tab                 | Gets the tab data to add.                                    |
| targetId (optional) | Gets the ID of the target tab to add the new tab.            |
| isAfter (optional)  | Defines whether the tab is added before or after the target. |

### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("addTab").Content("AddTab").Click("addTab").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Header("Home").Groups(group =>
{
group.Header("Clipboard").Collections(collection =>
{
collection.Items(item =>
{
item.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function addTab() {
var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
let newTab = {
header: "Insert",
id: "tab"
```

```

}
ribbonObj.addTab(newTab);
}
</script>

```

### addGroup

Using the **addGroup** method you can add a group dynamically in the ribbon control. The arguments are as follows.

| Argument name       | Description                                                    |
|---------------------|----------------------------------------------------------------|
| tabId               | Gets the ribbon tab ID.                                        |
| group               | Gets the group data to add.                                    |
| targetId (optional) | Gets the ID of the target group to add the new group.          |
| isAfter (optional)  | Defines whether the group is added before or after the target. |

### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("addGroup").Content("AddGroup").Click("addGroup").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("home").Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function addGroup() {
 var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
 let newGroup = {
 header: "newGroup",
 id: "insertGroup"
 }
 ribbonObj.addGroup("home", newGroup);
}
</script>

```

## addCollection

Using the `addCollection` method you can add a collection dynamically in the ribbon control. The arguments are as follows.

| Argument name       | Description                                                         |
|---------------------|---------------------------------------------------------------------|
| -----               | -----                                                               |
| groupId             | Gets the ribbon group ID.                                           |
| collection          | Gets the collection data to add.                                    |
| targetId (optional) | Gets the ID of the target collection to add the new collection.     |
| isAfter (optional)  | Defines whether the collection is added before or after the target. |

## INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("addCollection").Content("AddCollection").Click("addCollection").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Id("fontGroup").Header("Clipboard").Collections(collection =>
 {
 collection.Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
 }
 <script>
 function addCollection() {
 var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
 let newCollection = {
 id: "insertGroup",
 items: [
 {
 type: "Button",
 buttonSettings: {
 content: "Edit",
 iconCss: "e-icons e-edit"
 }
 },
 {
 type: "ColorPicker",
 colorPickerSettings: {
 value: "035a"
 }
 }
]
 }
 }
}
```

```

}
ribbonObj.addCollection("fontGroup", newCollection);
}
</script>

```

### addItem

Using the **addItem** method you can add an item dynamically in the ribbon control. The arguments are as follows.

| Argument name       | Description                                                   |
|---------------------|---------------------------------------------------------------|
| -----               | -----                                                         |
| collectionId        | Gets the ribbon collection ID.                                |
| item                | Gets the item data to add.                                    |
| targetId (optional) | Gets the ID of the target item to add the new item.           |
| isAfter (optional)  | Defines whether the item is added before or after the target. |

### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("addItem").Content("AddItem").Click("addItem").IsPrimary(
true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
 item.Type(RibbonItemType.Button).AllowedSizes(RibbonItemSize.Medium).ButtonS
ettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function addItem() {
 var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
 let newItem = {
 id: "insertItem",
 type: "ColorPicker",
 colorPickerSettings: {
 value: "035a"
 }
 }
 ribbonObj.addItem("buttonCollection", newItem);
}
</script>

```



### removeTab

The `removeTab` method is used to remove a tab from the ribbon control. This method takes the `tabId` as a parameter.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("removeTab").Content("RemoveTab").Click("removeTab").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("homeTab").Header("Home").Groups(group =>
 {
 group.Header("Clipboard").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 tab.Id("insertTab").Header("Insert").Add();
}).Render()
<script>
function removeTab() {
 var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
 ribbonObj.removeTab("insertTab");
}
</script>
```

### removeGroup

The `removeGroup` method is used to remove a group of items from the ribbon control. This method takes the `groupId` as a parameter.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("removeGroup").Content("RemoveGroup").Click("removeGroup").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("homeTab").Header("Home").Groups(group =>
 {
 group.Header("Font").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
 item.Type(RibbonItemType.Button).ButtonSettings(button =>
 {

```

```

button.IconCss("e-icons e-cut").Content("cut");
}).Add();
}).Add();
}).Add();
group.Header("ClipBoard").Id("clipBoard").Add();
}).Add();
}).Render()
<script>
function removeGroup() {
var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
ribbonObj.removeGroup("clipBoard");
}
</script>

```

### removeCollection

The `removeCollection` method is used to remove a collection of items from the ribbon control. This method takes the `collectionId` as a parameter.

### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("removeCollection").Content("RemoveCollection").Click("removeCollection").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Id("homeTab").Header("Home").Groups(group =>
{
group.Header("Font").Collections(collection =>
{
collection.Id("buttonCollection").Items(item =>
{
item.Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("cut");
}).Add();
}).Add();
collection.Id("colorPicker").Items(item =>
{
item.Type(RibbonItemType.ColorPicker).ColorPickerSettings(colorpicker =>
{
colorpicker.Value("035a");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function removeCollection() {
var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
ribbonObj.removeCollection("colorPicker");
}
</script>

```

### removeItem

The **removeItem** method is used to remove an item from the ribbon control. This method takes the **itemId** as a parameter.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("removeItem").Content("RemoveItem").Click("removeItem").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("homeTab").Header("Home").Groups(group =>
 {
 group.Header("Font").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
 item.Id("cutItem").Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 item.Id("copyItem").Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-copy").Content("copy");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
}).Render()
<script>
function removeItem() {
 var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
 ribbonObj.removeItem("copyItem");
}
</script>
```

### enableItem

The **enableItem** method in a ribbon control is used to enable a specific item in the ribbon control. This method takes the **itemId** as a parameter.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("enableItem").Content("EnableItem").Click("enableItem").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("homeTab").Header("Home").Groups(group =>
 {
 group.Header("Font").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
```

```

item.Id("cutItem").Disabled(true).Type(RibbonItemType.Button).ButtonSettings
(button =>
{
button.IconCss("e-icons e-cut").Content("cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function enableItem() {
var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
ribbonObj.enableItem("cutItem");
}
</script>

```

### disableItem

The **disableItem** method in a ribbon control is used to disable a specific item in the ribbon control. This method takes the **itemId** as a parameter.

#### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("disableItem").Content("DisableItem").Click("disableItem")
).IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
tab.Id("homeTab").Header("Home").Groups(group =>
{
group.Header("Font").Collections(collection =>
{
collection.Id("buttonCollection").Items(item =>
{
item.Id("cutItem").Type(RibbonItemType.Button).ButtonSettings(button =>
{
button.IconCss("e-icons e-cut").Content("cut");
}).Add();
}).Add();
}).Add();
}).Add();
}).Render()
<script>
function disableItem() {
var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
ribbonObj.disableItem("cutItem");
}
</script>

```

### refreshLayout

The **refreshLayout** method can be used to update the layout.

#### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon

```

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("refresh").Content("Refresh").Click("refreshLayout").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("homeTab").Header("Home").Groups(group =>
 {
 group.Header("Font").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
 item.Id("cutItem").Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Render()
<script>
function refreshLayout() {
 var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
 ribbonObj.activeLayout = "Simplified";
 ribbonObj.refreshLayout();
}
</script>

```

### selectTab

The **selectTab** method is used to select a tab from the ribbon control. This method takes the **tabId** as a parameter.

### INDEX.CSHTML

```

@using Syncfusion.EJ2.Ribbon
@using Syncfusion.EJ2.Navigations
@Html.EJS().Button("selectTab").Content("selectTab").Click("selectTab").IsPrimary(true).Render()
@Html.EJS().Ribbon("ribbon").Tabs(tab =>
{
 tab.Id("homeTab").Header("Home").Groups(group =>
 {
 group.Header("Font").Collections(collection =>
 {
 collection.Id("buttonCollection").Items(item =>
 {
 item.Id("cutItem").Type(RibbonItemType.Button).ButtonSettings(button =>
 {
 button.IconCss("e-icons e-cut").Content("cut");
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 }).Add();
 tab.Id("insertTab").Header("Insert").Add();
 }).Render()
<script>
function selectTab() {

```

```
var ribbonObj = document.getElementById("ribbon").ej2_instances[0];
ribbonObj.selectTab("insertTab");
}
</script>
```

## RichTextEditor

### Getting Started with ASP.NET MVC Rich Text Editor Control

This section briefly explains about how to include [ASP.NET MVC Rich Text Editor](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

**~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

**Register Syncfusion script manager**

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

**Add ASP.NET MVC Rich Text Editor control**

Now, add the Syncfusion ASP.NET MVC Rich Text Editor control in `~/Views/Home/Index.cshtml` page.

**CSHTML**

```
@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<div>
 <div>
 <p>The Rich Text Editor is WYSIWYG ('what you see is what you get')
editor useful to create and edit content, and return the valid HTML
markup or markd
own of the content</p>
 <p> Toolbar </p>

 <p> Toolbar contains commands to align the text, insert
link, insert image, insert list, undo / redo operations, HTML view, etc </p>

 <p> Toolbar is fully customizable </p>

 <p> Links </p>


```

```

 <p> You can insert a hyperlink with its corresponding
dialog</p>

 <p> Attach a hyperlink to the displayed text. </p>

 <p> Customize the quick toolbar based on the hyperlink </p>

 <p> Image.</p>

 <p> Allows you to insert images from an online source as
well as the local computer</p>

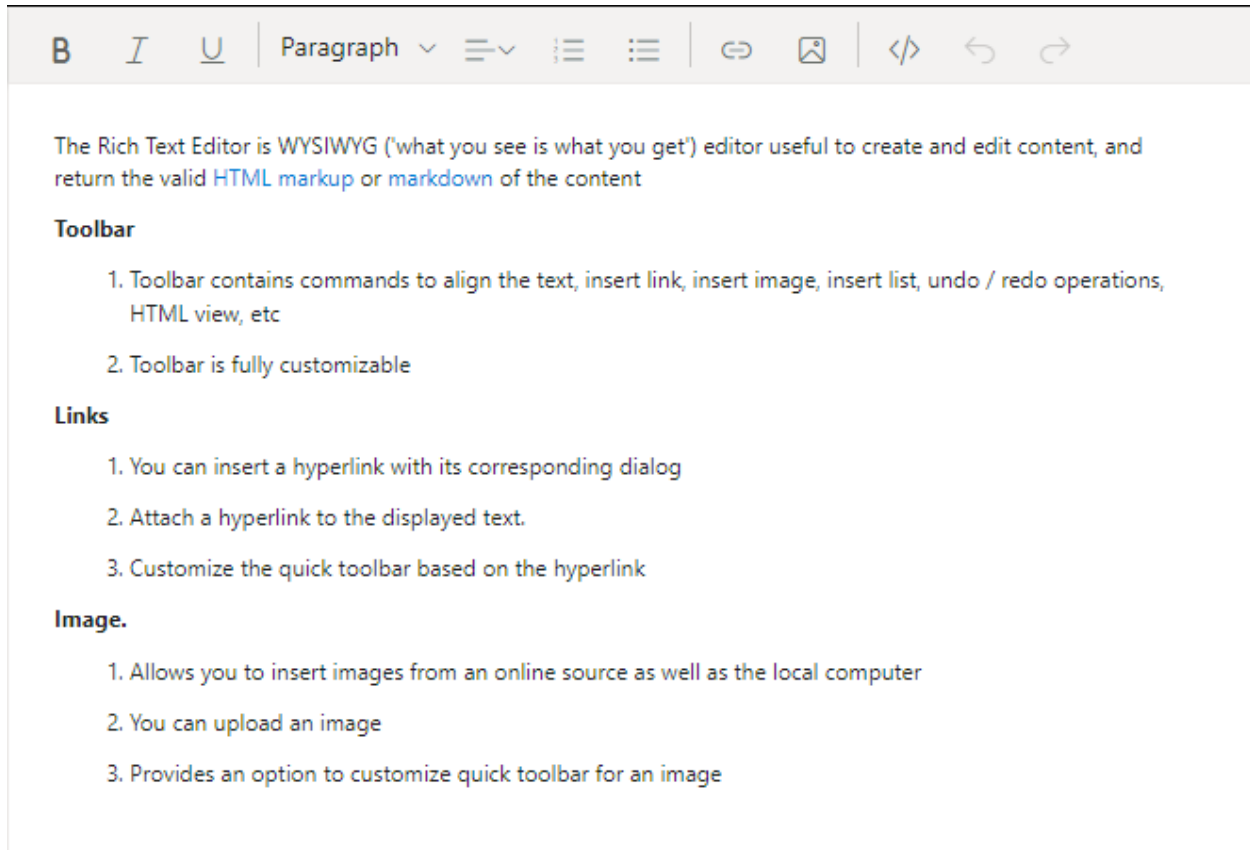
 <p> You can upload an image</p>

 <p> Provides an option to customize quick toolbar for an
image </p>

</div>
</div>).Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Rich Text Editor control will be rendered in the default web browser.





#### Initialize from iframe element

Initialize the Rich Text Editor on `<div>` element as shown below and set the enable field of [IframeSettings](#) property as true to render the Rich Text Editor content in an `<iframe>` and its isolated from the rest of the page.

#### CSHTML

```
@Html.EJS().RichTextEditor("iframe").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>

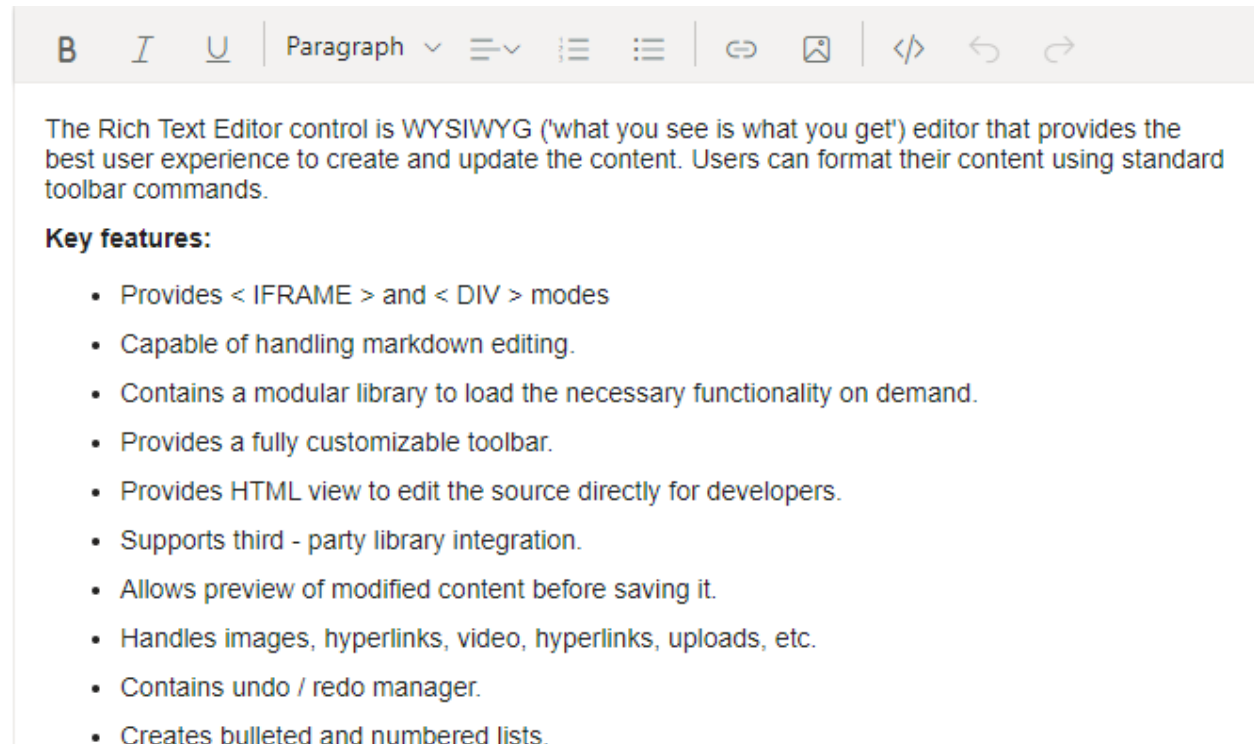
</div>
```

```

<p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).IframeSettings(iframeSettings =>
iframeSettings.Enable(true)).Render()

```



The screenshot shows the Rich Text Editor control. At the top is a toolbar with icons for Bold (B), Italic (I), Underline (U), Paragraph (dropdown), Bulleted List, Numbered List, Link, Image, Source Code, Undo, and Redo. Below the toolbar is the content area, which contains the following text:

The Rich Text Editor control is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.

**Key features:**

- Provides < IFRAME > and < DIV > modes
- Capable of handling markdown editing.
- Contains a modular library to load the necessary functionality on demand.
- Provides a fully customizable toolbar.
- Provides HTML view to edit the source directly for developers.
- Supports third - party library integration.
- Allows preview of modified content before saving it.
- Handles images, hyperlinks, video, hyperlinks, uploads, etc.
- Contains undo / redo manager.
- Creates bulleted and numbered lists.

### Configure the Toolbar

Configure the toolbar with the tools using items field of the [ToolbarSettings](#) property as your application requires.

### CSHTML

```

@model List<string>
@Html.EJS().RichTextEditor("toolbar").ToolbarSettings(e =>
e.Items((object)Model)).ContentTemplate(@<div>
<p>
The Rich Text Editor control is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
Users can format their content using standard toolbar commands.
</p>
<p> Key features:</p>

<p> Provides < IFRAME > and < DIV > modes
</p>
<p> Capable of handling markdown editing.</p>
<p> Contains a modular library to load the necessary
functionality on demand.</p>
<p> Provides a fully customizable toolbar.</p>

```

```

<p> Provides HTML view to edit the source directly for
developers.</p>
<p> Supports third - party library integration.</p>
<p> Allows preview of modified content before saving
it.</p>
<p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).Render()

```

## HOMECONTROLLER.CS

```

public ActionResult Index()
{
 List<string> tools = new List<string>() {
 "Bold", "Italic", "Underline", "StrikeThrough",
 "FontName", "FontSize", "FontColor", "BackgroundColor",
 "LowerCase", "UpperCase", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList",
 "Outdent", "Indent", "|",
 "CreateLink", "Image", "CreateTable", "|", "ClearFormat", "Print",
 "SourceCode", "FullScreen", "|", "Undo", "Redo"
 };
 return View(tools);
}

```

B
I
U
↺
Segoe UI
10 pt
A
↺
↻
↺
↻
tt
Tt
Paragraph
...

The Rich Text Editor control is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.

**Key features:**

- Provides < IFRAME > and < DIV > modes
- Capable of handling markdown editing.
- Contains a modular library to load the necessary functionality on demand.
- Provides a fully customizable toolbar.
- Provides HTML view to edit the source directly for developers.
- Supports third - party library integration.
- Allows preview of modified content before saving it.
- Handles images, hyperlinks, video, hyperlinks, uploads, etc.
- Contains undo / redo manager.
- Creates bulleted and numbered lists.

**Note:** `|` and `-` can insert a vertical and horizontal separator lines in the toolbar.

### Insert Images and Links

The **Image** module inserts an image into Rich Text Editor's content area, and the **Link** module links external resources such as website URLs, to selected text in the Rich Text Editor's content, respectively.

The link inject module adds a link icon to the toolbar and the image inject module adds an image icon to the toolbar.

Specifies the items to be rendered in quick toolbar based on the target element such image, link and text element. The quick toolbar opens to customize the element by clicking the target element.

### CSHTML

```
@model List<string>
@Html.EJS().RichTextEditor("image").ToolbarSettings(e =>
e.Items((object)Model)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).QuickToolbarSettings(e => { e.Image((object)ViewBag.image);
}).Render()
```

### HOMECONTROLLER.CS

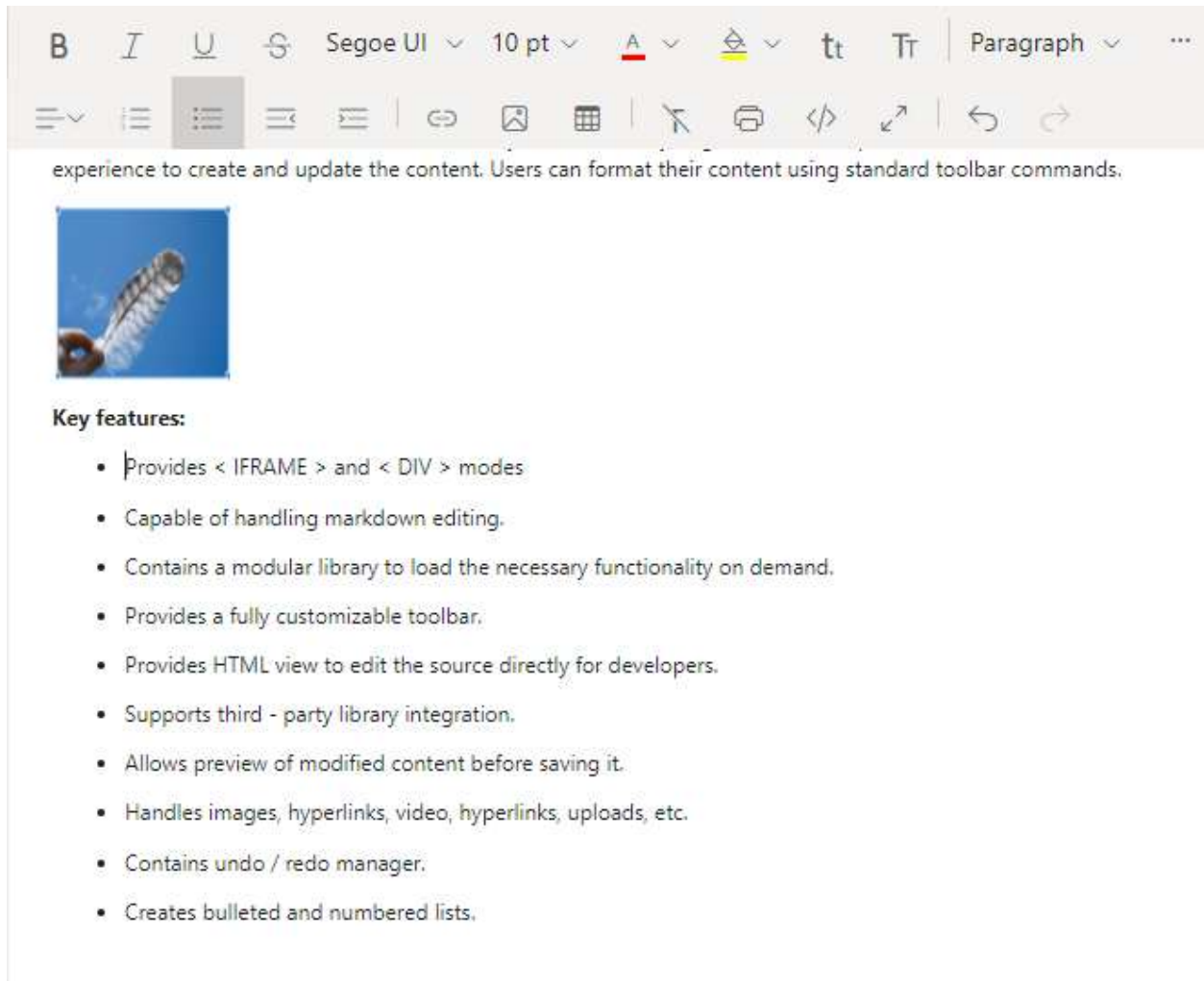
```
public ActionResult Index()
{
 ViewBag.image = new[] {
 "Replace", "Align", "Caption", "Remove", "InsertLink", "OpenImageLink", "-",
 "EditImageLink", "RemoveImageLink", "Display", "AltText", "Dimension"
 };
 List<string> tools = new List<string>() {
 "Bold", "Italic", "Underline", "StrikeThrough",
 "FontName", "FontSize", "FontColor", "BackgroundColor",
 "LowerCase", "UpperCase", "|",

```

```

"Formats", "Alignments", "OrderedList", "UnorderedList",
"Outdent", "Indent", "|",
"CreateLink", "Image", "CreateTable", "|", "ClearFormat", "Print",
"SourceCode", "FullScreen", "|", "Undo", "Redo"
};
return View(tools);
}

```



### Send formatted content using XmlHttpRequest

The Html string of the Rich Text Editor can be passed from View to the Controller through the **XMLHttpRequest Post** action. The HTML value binds to the corresponding mapped controller, and you can access it in the Post action parameter.

### CSHTML

```

@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<div>
 <div>
 <p>The Rich Text Editor is WYSIWYG ('what you see is what you get')
 editor useful to create and edit content, and return the valid HTML
 markup or <a

```

```

href='https://ej2.syncfusion.com/aspnetmvc/RichTextEditor/DefaultMode'>markd
own of the content</p>
 <p> Toolbar </p>

 <p> Toolbar contains commands to align the text, insert
link, insert image, insert list, undo / redo operations, HTML view, etc </p>

 <p> Toolbar is fully customizable </p>

 <p> Links </p>

 <p> You can insert a hyperlink with its corresponding
dialog</p>

 <p> Attach a hyperlink to the displayed text. </p>

 <p> Customize the quick toolbar based on the hyperlink </p>

 <p> Image.</p>

 <p> Allows you to insert images from an online source as
well as the local computer</p>

 <p> You can upload an image</p>

 <p> Provides an option to customize quick toolbar for an
image </p>

</div>
</div>).Render()
<button id="getRTEvalue">Get RichTextEditor Value</button>
<script>
 document.getElementById('getRTEvalue').addEventListener('click',
function () {
 var obj = document.getElementById('defaultRTE').ej2_instances[0];
 var value = JSON.stringify(
 { text: obj.value }
);
 const Http = new XMLHttpRequest();
 const url = '@Url.Action("Save")';
 Http.open("POST", url);
 Http.setRequestHeader('Content-Type', 'application/json');
 Http.send(value);
});
</script>

```

### Retrieve the Formatted Content

To retrieve the editor contents, use [value](#) property of Rich Text Editor.

```
`javascript
var rteValue = this.rteObj.value;
`
```

Or, you can use the public method, `getHtml` to retrieve the Rich Text Editor content.

```
`javascript
var rteValue = this.rteObj.getHtml();
`
```

To fetch the Rich Text Editor's text content, use `getText` method of Rich Text Editor.

```
`javascript
var rteValue = this.rteObj.getText();
`
```

### Retrieve the number of characters

To get the maximum number of characters in the Rich Text Editor's content, use `getCharCount`

```
`typescript
let rteCount = this.rteObj.getCharCount();
`
```

The final output will be displayed as follows

### CSHTML

```
@model List<string>
@Html.EJS().RichTextEditor("link").ToolbarSettings(e =>
e.Items((object)Model)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>

</div>
```

```

<p> Allows preview of modified content before saving
it.</p>
<p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).Render()

```

## HOMECONTROLLER.CS

```

public ActionResult Index()
{
 List<string> tools = new List<string>() {
 "Bold", "Italic", "Underline", "StrikeThrough",
 "FontName", "FontSize", "FontColor", "BackgroundColor",
 "LowerCase", "UpperCase", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList",
 "Outdent", "Indent", "|",
 "CreateLink", "Image", "CreateTable", "|", "ClearFormat", "Print",
 "SourceCode", "FullScreen", "|", "Undo", "Redo"
 };
 return View(tools);
}

```

**Note:** [View Sample in GitHub.](#)

See also

- [Real time example using Rich Text Editor](#)
- [How to insert Emoticons](#)
- [How to change the editor type](#)
- [How to render the iframe](#)
- [How to render the toolbar in inline mode](#)

**Note:** You can refer to our [ASP.NET MVC Rich Text Editor](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Rich Text Editor example](#) that shows how to render the rich text editor tools.

## Formation

The Rich Text Editor control used to create and edit the content and return valid HTML markup or markdown (MD) of the content. It supports the following two editing formation.

- HTML Editor
- Markdown Editor

## HTML Editor

Rich Text Editor is a WYSIWYG editing control for formatting the word content as HTML.

The HTML editing mode is the default mode in Rich Text Editor to format the content through the available toolbar items to return the valid HTML markup. Set the [EditorMode](#) property as **HTML**.



**CSHTML**

```
@Html.EJS().RichTextEditor("htmlEditor").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

## Markdown Editor

Set the [EditorMode](#) property as Markdown, to create or edit the content and apply formatting to view markdown formatted content.

- The Supported Tags are h6,h5,h4,h3,h2,h1,blockquote,pre,p,ol,ul.
- The Supported Selection Tags are Bold, Italic, StrikeThrough, InlineCode, SubScript, SuperScript, UpperCase, LowerCase.
- The supported insert commands are Image, Link and Table.

**Note:** The third-party library such as [Marked](#) or any other library is used to convert markdown into HTML content.

**CSHTML**

```
@using Syncfusion.EJ2.RichTextEditor
```

```
@Html.EJS().RichTextEditor("markdown").Value((string)ViewBag.value).EditorMode(EditorMode.Markdown).Render()
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.value = @"""**Overview**
 The Rich Text Editor control is WYSIWYG ('what you see is what you get')
 editor used to create and edit the content and return valid HTML markup or
 markdown (MD) of the content. The editor provides a standard toolbar to
 format content using its commands. Modular library features to load the
 necessary functionality on demand. The toolbar contains commands to align
 the text, insert link, insert image, insert list, undo/redo operation, HTML
 view, and more.
 Key features
 - *Mode*: Provides IFRAME and DIV mode.
 - *Module*: Modular library to load the necessary functionality on demand.
 - *Toolbar*: Provide a fully customizable toolbar.
 - *Editing*: HTML view to edit the source directly for developers.
 - *Third-party Integration*: Supports to integrate third-party library.
 - *Preview*: Preview the modified content before saving it.
 - *Tools*: Handling images, hyperlinks, video, uploads and more.
 - *Undo and Redo*: Undo/redo manager.
 - *Lists*:Creates bulleted and numbered list.";
 return View();
 }
}
```

## See Also

- [How to integrate the third party library](#)
- [How to render the iframe](#)

## Toolbar

The Rich Text Editor's toolbar contains a collection of tools such as bold, italic and text alignment buttons that are used to format the content. However, in most integrations, it's desirable to change the toolbar configuration to suit needs. Fortunately, that's quite easy to do too.

To create Rich Text Editor with Markdown editing feature, it can be added by defining the Toolbar as a collection of built-in items.

The Rich Text Editor allows you to configure different types of toolbar using **Type** field in [ToolbarSettings](#) property. The types of toolbar are:

1. Expand
2. MultiRow

### Expand Toolbar

The default mode of [Type](#) is **Expand**, it will hide the overflowing items in the next row. By clicking the expand arrow, view the overflowing toolbar items.

#### CSHTML

```
@using Syncfusion.EJ2.RichTextEditor
<div style="width:500px; margin:0 auto;">
 @Html.EJS().RichTextEditor("toolbar-types").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

 </div>).ToolbarSettings(e => { e.Type(ToolbarType.Expand); }).Render()
</div>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Multi-row Toolbar

Set the type as **MultiRow** in [ToolbarSettings](#) to moves the overflowing items in the next row. All toolbar items are visible.

#### CSHTML

```
@using Syncfusion.EJ2.RichTextEditor
<div style="width:500px; margin:0 auto;">
 @Html.EJS().RichTextEditor("toolbar-types").ContentTemplate(@<div>
```

```

<p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
</p>
<p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
</p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).ToolbarSettings(e => { e.Type(ToolbarType.MultiRow); }).Render()
</div>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Floating Toolbar

By default, toolbar is float at the top of the Rich Text Editor on scrolling. By disabling the [EnableFloating](#), toolbar at the top of the content area of Rich Text Editor. Floating toolbar offset can be customize by specifies the offset of the floating toolbar from documents top position using [FloatingToolbarOffset](#).

Enable or disable the floating toolbar using `EnableFloating` of `ToolbarSettings` property.

## CSHTML

```

@using Syncfusion.EJ2.RichTextEditor
<div class="control-wrapper">
 <div class="control-section">
 @Html.EJS().RichTextEditor("toolbar-
 floating").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is
 what you get') editor that provides the best user experience to create and
 update the content.
 </p>

```

```

 Users can format their content using standard toolbar
 commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV >
 modes </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks,
 uploads, etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

 </div>).Created("created").Height("720px").ToolbarSettings(e => {
 e.Type(ToolbarType.Expand).EnableFloating(false); }).Render()
 </div>
 <div class="col-lg-4">
 Enable/Disable Floating:
 @Html.EJS().CheckBox("float").Checked(false).Label("Enable
 Floating").Change("checkchange").Render()
 </div>
</div>
<script type="text/javascript">
 var defaultRTE;
 function created() {
 defaultRTE = this;
 }
 function checkchange(args) {
 defaultRTE.toolbarSettings.enableFloating = args.checked;
 defaultRTE.dataBind();
 }
</script>

```

## CONTROLLER.CS

```















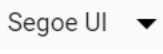


public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}


```

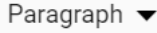
## Toolbar Items


The following table lists the available tools on the toolbar.


| Name | Icons | Summary | Initialization |
|------|-------|---------|----------------|
|------|-------|---------|----------------|


|                         |                                                                                     |                                                                                                                                                   |
|-------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| ----- ----- ----- ----- |                                                                                     |                                                                                                                                                   |
| Undo                    |    | Allows to undo the actions.  toolbarSettings: { items: ['Undo']}                                                                                  |
| Redo                    |    | Allows to redo the actions.  toolbarSettings: { items: ['Redo']}                                                                                  |
| Alignment               |    | Align the content with left, center, and right margin.  toolbarSettings: { items: ['Alignments']}                                                 |
| OrderedList             |    | Create a new list item(numbered).  toolbarSettings: { items: ['OrderedList']}                                                                     |
| UnorderedList           |    | Create a new list item(bulleted).  toolbarSettings: { items: ['UnorderedList']}                                                                   |
| Indent                  |    | Allows to increase the indent level of the content.  toolbarSettings: { items: ['Indent']}                                                        |
| Outdent                 |    | Allows to decrease the indent level of the content.  toolbarSettings: { items: ['Outdent']}                                                       |
| Hyperlink               |    | Creates a hyperlink to a text or image to a specific location in the content.  toolbarSettings: { items: ['CreateLink']}                          |
| Images                  |   | Inserts an image from an online source or local computer.  toolbarSettings: { items: ['Image']}                                                   |
| LowerCase               |  | Change the case of selected text to lower in the content.  toolbarSettings: { items: ['LowerCase']}                                               |
| UpperCase               |  | Change the case of selected text to upper in the content.  toolbarSettings: { items: ['UpperCase']}                                               |
| SubScript               |  | Makes the selected text as subscript (lower).  toolbarSettings: { items: ['SubScript']}                                                           |
| SuperScript             |  | Makes the selected text as superscript (higher).  toolbarSettings: { items: ['SuperScript']}                                                      |
| Print                   |  | Allows to print the editor content.  toolbarSettings: { items: ['Print']}                                                                         |
| FontName                |  | Defines the fonts that appear under the Font Family DropDownList from the Rich Text Editor's toolbar.  toolbarSettings: { items: ['FontName']}    |
| FontSize                |  | Defines the font sizes that appear under the Font Size DropDownList from the Rich Text Editor's toolbar.  toolbarSettings: { items: ['FontSize']} |
| FontColor               |  | Specifies an array of colors can be used in the colors popup for font color.  toolbarSettings: { items: ['FontColor']}                            |


|  | Specifies an array of colors can be used in the colors popup for background color. | toolbarSettings: { items: ['BackgroundColor']} |


|  | An Object with the options that will appear in the Paragraph Format dropdown from the toolbar. | toolbarSettings: { items: ['Formats']} |


|  | Apply double line strike through formatting for the selected text. | toolbarSettings: { items: ['StrikeThrough']} |


|  | The clear format tool is useful to remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. As a result, all the text formatting will be cleared and return to its default formatting styles. | toolbarSettings: { items: ['ClearFormat']} |


|  | Stretches the editor to the maximum width and height of the browser window. | toolbarSettings: { items: ['FullScreen']} |


|  | Rich Text Editor includes the ability for users to directly edit HTML code via “Source View”. If you made any modification in Source view directly, synchronize with Design view. | toolbarSettings: { items: ['SourceCode']} |


|  | Allows to create list items with various list style types(numbered). | toolbarSettings: { items: ['NumberFormatList']} |


|  | Allows to create list items with various list style types(bulleted). | toolbarSettings: { items: ['BulletFormatList']} |


|  | Allows each line to begin at the same distance from the editor’s left-hand side. | toolbarSettings: { items: ['JustifyLeft']} |


|  | There is an even space on each side of each line since the text is not aligned to the left or right margins. | toolbarSettings: { items: ['JustifyCenter']} |


|  | Allows each line to end at the same distance from the editor’s right-hand side. | toolbarSettings: { items: ['JustifyRight']} |


|  | The text is aligned with both right and left margins. | toolbarSettings: { items: ['JustifyFull']} |


|  | Text that is thicker and darker than usual. | toolbarSettings: { items: ['Bold']} |


|  | Shows a text that is leaned to the right. | toolbarSettings: { items: ['Italic']} |


|  | The underline is added to the selected text. | toolbarSettings: { items: ['Underline']} |


| ClearAll |  | Removes all styles that have been applied to the selected text. | toolbarSettings: { items: ['ClearAll']} |


| Cut |  | Removes the text from its current location and places it into the clipboard. | toolbarSettings: { items: ['Cut']} |


| Copy |  | The selected item is copied and pasted into the clipboard. | toolbarSettings: { items: ['Copy']} |


| Paste |  | Allows you to insert a clipboard item into a specific location. | toolbarSettings: { items: ['Paste']} |


| OpenLink |  | To open the URL link that is attached to the selected text. | toolbarSettings: { items: ['OpenLink']} |


| EditLink |  | Allows you to change the URL that has been attached to a specific item. | toolbarSettings: { items: ['EditLink']} |


| CreateTable |  | Create a table with defined columns and rows. | toolbarSettings: { items: ['CreateTable']} |


| RemoveTable |  | Removes the selected table and its contents. | toolbarSettings: { items: ['TableRemove']} |


| Replace |  | Replace the selected image with another image. | toolbarSettings: { items: ['Replace']} |

| Align |  | The image can be aligned to the right, left, or center. | toolbarSettings: { items: ['Align']} |


| Remove |  | Allows to remove the selected image from the editor. | toolbarSettings: { items: ['Remove']} |


| OpenImageLink |  | Opens the link that is attached to the selected image. | toolbarSettings: { items: ['OpenImageLink']} |


| EditImageLink |  | Allows to edit the link that is attached to the selected image. | toolbarSettings: { items: ['EditImageLink']} |


| RemoveImageLink |  | Removes the link that is attached to the selected image. | toolbarSettings: { items: ['RemoveImageLink']} |





| InsertLink |  | Allows users to add a link to a particular item. | toolbarSettings: { items: ['InsertLink']} |


| Display |  | Allows you to choose whether an image should be shown inline or as a block. | toolbarSettings: { items: ['Display']} |


| AltText |  | To display image description when an image on a Web page cannot be displayed. | toolbarSettings: { items: ['AltText']} |


| Dimension |  | Allows you to customize the image's height and width. | toolbarSettings: { items: ['Dimension']} |

| Maximize |  | Stretches the editor to the maximum width and height of the browser window. | toolbarSettings: { items: ['Maximize']} |


| Minimize |  | Shrinks the editor to the default width and height. | toolbarSettings: { items: ['Minimize']} |


| Preview |  | Allows to see how the editor's content looks in a browser. | toolbarSettings: { items: ['Preview']} |


| InsertCode |  | Represents preformatted text which is to be presented exactly as written in the HTML file. | toolbarSettings: { items: ['InsertCode']} |


| RemoveLink |  | Allows you to remove the applied link from the selected item. | toolbarSettings: { items: ['RemoveLink']} |


| TableHeader |  | Allows you to add a table header. | toolbarSettings: { items: ['TableHeader']} |

| TableColumns |  | Shows the dropdown to insert a column or delete the selected column. | toolbarSettings: { items: ['TableColumns']} |

| TableRows |  | Shows the dropdown to insert a row or delete the selected row. | toolbarSettings: { items: ['TableRows']} |

| TableCellHorizontalAlign |  | Allows the table cell content to be aligned horizontally. | toolbarSettings: { items: ['TableCellHorizontalAlign']} |

| TableCellVerticalAlign |  | Allows the table cell content to be aligned vertically. | toolbarSettings: { items: ['TableCellVerticalAlign']} |

| TableEditProperties |  | Allows you to change the table width, padding, and cell spacing styles. | toolbarSettings: { items: ['TableEditProperties']} |

By default, tool will be arranged in following order.

**Note:** items: ['Bold', 'Italic', 'Underline', '|', 'Formats', 'Alignments', 'OrderedList', 'UnorderedList', '|', 'CreateLink', 'Image', '|', 'SourceCode', 'Undo', 'Redo']

We can customize the tools order as our application requirement. If you are not specifying any tools order, the editor will create the toolbar with default items.

### Custom Tool

The Rich Text Editor allows you to configure your own commands to its toolbar using [ToolbarSettings](#) property. The command can be plain text, icon, or HTML template. You can also define the order and group where the command should be included. Bind the action to the command by getting its instance.

This sample shows how to add your own commands to toolbar of the Rich Text Editor. The “Ω” command is added to insert special characters in the editor.

The below code snippet, for custom tool with tooltip text which will be included in **Items** field of **ToolbarSettings** property.

```
`csharp
var tools = new {
 tooltipText = "Insert Symbol",

 template = "<button class='e-tbar-btn e-btn' tabindex='-1' id='custom_tbar' style='width:100%'><div
class='e-tbar-btn-text rtecustomtool' style='font-weight: 500;'> Ω</div></button>"

};

ViewBag.items = new object[] { "Bold", "Italic", "Underline", "|", "Formats", "Alignments", "OrderedList",
"UnorderedList", "|", "CreateLink", "Image", "|", "SourceCode", tools
, "|", "Undo", "Redo"
};
`
```

Click the “Ω” command to show the special characters list, and then choose the character to be inserted in the editor.

### CSHTML

```
<div id="rteSection">
 @Html.EJS().RichTextEditor("customtool").ContentTemplate(@<div><p>
 The custom command 'insert special character' is configured as the last
 item of the toolbar.
 Click on the command and choose the special character you want to
 include from the popup.
 </p></div>).Created("created").ToolbarSettings(e =>
 e.Items((object)ViewBag.items)).Render()
 @Html.EJS().Dialog("customTbarDialog").Visible(false).Header("Special
 Characters").Created("onDialogCreate").ZIndex(1000).ShowCloseIcon(false).IsM
 odal(true).cssClass('e-rte-
 elements').OverlayClick("dialogOverlay").Buttons(e =>
 {
 e.ButtonModel(ViewBag.button).Click("onInsert").Add();
 e.ButtonModel(ViewBag.button1).Click("dialogOverlay").Add();
 })
</div>
```

```

 }).ContentTemplate(@<div id="rteSpecial_char">
 <div class="char_block" title="^">^</div>
 <div class="char_block" title=" "> </div>
 <div class="char_block" title="`">`</div>
 <div class="char_block" title="{ ">{</div>
 <div class="char_block" title="|">|</div>
 <div class="char_block" title="}">}</div>
 <div class="char_block" title="~">~</div>
 <div class="char_block" title=" "> </div>
 <div class="char_block" title=";">;</div>
 <div class="char_block" title="ç">ç</div>
 <div class="char_block" title="£">£</div>
 <div class="char_block" title="¤">¤</div>
 <div class="char_block" title="¥">¥</div>
 <div class="char_block" title="₹">₹</div>
 <div class="char_block" title="|">|</div>
 <div class="char_block" title="$">$</div>
 <div class="char_block" title="...">...</div>
 <div class="char_block" title="©">©</div>
 <div class="char_block" title="ª">ª</div>
 <div class="char_block" title="«">«</div>
 <div class="char_block" title="¬">¬</div>
 <div class="char_block" title="–">–</div>
 <div class="char_block" title="®">®</div>
 <div class="char_block" title="¯">¯</div>
 <div class="char_block" title="°">°</div>
 <div class="char_block" title="±">±</div>
 <div class="char_block" title="²">²</div>
 <div class="char_block" title="³">³</div>
 <div class="char_block" title="´">´</div>
 <div class="char_block" title="µ">µ</div>
 <div class="char_block" title="¶">¶</div>
 <div class="char_block" title="·">·</div>
 <div class="char_block" title="¸">¸</div>
 <div class="char_block" title="¹">¹</div>
 <div class="char_block" title="º">º</div>
 <div class="char_block" title="»">»</div>
 <div class="char_block" title="¼">¼</div>
 <div class="char_block" title="½">½</div>
 <div class="char_block" title="¾">¾</div>
 <div class="char_block" title="¿">¿</div>
 <div class="char_block" title="À">À</div>
 <div class="char_block" title="Á">Á</div>
 <div class="char_block" title="Â">Â</div>
 <div class="char_block" title="Ã">Ã</div>
 </div>).Render()
</div>
<script>
 var defaultRTE, selection, ranges, dialog;
 function onDialogCreate() {
 dialog = this;
 dialog.element.style.maxHeight = '300px';
 }
 function onInsert() {
 var activeEle = dialog.element.querySelector('.char_block.e-active');
 if (defaultRTE.formatter.getUndoRedoStack().length === 0) {

```

```

 defaultRTE.formatter.saveData();
 }
 defaultRTE.executeCommand('insertText', activeEle.textContent);
 defaultRTE.formatter.saveData();
 defaultRTE.formatter.enableUndo(defaultRTE);
 dialogOverlay();
}
function dialogOverlay() {
 var activeEle = dialog.element.querySelector('.char_block.e-
active');
 if (activeEle) {
 activeEle.classList.remove('e-active');
 }
 dialog.hide();
}
function created() {
 defaultRTE = this;
 selection = new ej.richtexteditor.NodeSelection();
 var customBtn = defaultRTE.element.querySelector('#custom_tbar');
 customBtn.onclick = function (e) {
 dialog.element.style.display = '';
 ranges = selection.getRange(document);
 dialog.width = defaultRTE.element.offsetWidth * 0.5;
 dialog.dataBind();
 dialog.show();
 var dialogCtn = document.getElementById('rteSpecial_char');
 dialogCtn.onclick = function (e) {
 var target = e.target;
 var activeEle = dialog.element.querySelector('.char_block.e-
active');
 if (target.classList.contains('char_block')) {
 target.classList.add('e-active');
 if (activeEle) {
 activeEle.classList.remove('e-active');
 }
 }
 };
 customBtn.onclick = function (e) {
 defaultRTE.focusIn();
 dialog.element.style.display = '';
 ranges = selection.getRange(document);
 dialog.width = defaultRTE.element.offsetWidth * 0.5;
 dialog.dataBind();
 dialog.show();
 };
 };
}
</script>
<style>
#customTbarDialog #special_char,
#customTbarDialog .char_block {
 display: inline-block;
}
#custom_tbar,
#custom_tbar div {
 cursor: pointer;
}

```

```

#rteSpecial_char {
 padding: 15px 0 15px 0;
}
#customTbarDialog .char_block.e-active {
 outline: 1px solid #e3165b;
 border-color: #e3165b;
}
#customTbarDialog .char_block {
 width: 30px;
 height: 30px;
 line-height: 30px;
 margin: 0 5px 5px 0;
 text-align: center;
 vertical-align: middle;
 border: 1px solid #DDDDDD;
 font-size: 20px;
 cursor: pointer;
 user-select: none;
}
#custom_tbar .rtcustomtool {
 font-size: 18px;
}
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 var tools = new
 {
 tooltipText = "Insert Symbol",
 template = "<button class='e-tbar-btn e-btn' tabindex='-1'
id='custom_tbar' style='width:100%'><div class='e-tbar-btn-text
rtcustomtool' style='font-weight: 500;'> Ω</div></button>"
 };
 ViewBag.items = new object[] { "Bold", "Italic", "Underline", "|",
"Formats", "Alignments", "OrderedList",
"UnorderedList", "|", "CreateLink", "Image", "CreateTable", "|",
"SourceCode", tools
, "|", "Undo", "Redo"
 };
 ViewBag.button = new
 {
 content = "Insert",
 isPrimary = true
 };
 ViewBag.button1 = new
 {
 content = "Cancel"
 };
 return View();
 }
}

```

The focus will be lost while rendering the required component for the custom toolbar, causing it to render outside the Rich Text Editor and triggering a blur event. During that time, proper functionality will not be achievable. Therefore, it is recommended to set the `cssClass` property or class as `e-rte-elements` in the dependency component.

### Quick Inline Toolbar

Quick commands are opened as popup on clicking the corresponding element. The commands must be passed as string collection to image, text, and link attributes of the [QuickToolbarSettings](#) property.

| Target Element | Default Quick Toolbar items |

|-----|-----|

| Image | 'Replace', 'Align', 'Caption', 'Remove', 'InsertLink', 'Display', 'AltText', 'Dimension'.|

| Link | 'Open', 'Edit', 'UnLink'.|

| Text | null <br> (Any toolbar [items](#) in the Rich Text Editor can be configured here).|

| Table| 'TableHeader', 'TableRows', 'TableColumns', 'BackgroundColor', '-', 'TableRemove', 'Alignments', 'TableCellVerticalAlign', 'Styles'.|

Custom tool can be added to the corresponding quick toolbar, using [QuickToolbarSettings](#) property.

The below sample demonstrates the option to insert the image to the Rich Text Editor content as well as option to rotate the image through the quick toolbar.

### CSHTML

```
<div class="control-section">

@Html.EJS().RichTextEditor("inline").Created("onCreated").ToolbarClick("onToolbarClick").InlineMode(e =>
e.Enable(true).OnSelection(true)).ToolbarSettings(e =>
e.Items((object)ViewBag.items)).QuickToolbarSettings(e => {
e.Image((object)ViewBag.image); }).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes</p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
 <p>Contains undo / redo manager.</p>

</div>
```

```

 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
</div>
<style>
 .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-left::before {
 content: "\e341";
 }
 .e-rte-quick-popup .e-rte-quick-toolbar .e-rotate-right::before {
 content: "\e354";
 }
</style>
<script>
 function onCreate() {
 defaultRTE = this;
 }
 function onToolbarClick(e) {
 var nodeObj = new ej.richtexteditor.NodeSelection();
 var range =
nodeObj.getRange(defaultRTE.contentModule.getDocument());
 var imgEle = nodeObj.getNodeCollection(range)[0];
 var transform;
 if (e.item.tooltipText === 'Rotate Right') {
 transform = (imgEle.style.transform === '') ? 0 :
 parseInt(imgEle.style.transform.split('(')[1].split(' ')[0],
10);
 imgEle.style.transform = 'rotate(' + (transform + 90) + 'deg)';
 }
 else if (e.item.tooltipText === 'Rotate Left') {
 transform = (imgEle.style.transform === '') ? 0 :
Math.abs(parseInt(imgEle.style.transform.split('(')[1].split(' ')[0], 10));
 imgEle.style.transform = 'rotate(-' + (transform + 90) + 'deg)';
 }
 }
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Bold", "Italic", "Underline",
 "Formats", "Alignments", "-", "OrderedList", "UnorderedList",
 "Image",
 "CreateLink" };
 object tools1 = new
 {
 tooltipText = "Rotate Left",
 template = "<button class='e-tbar-btn e-btn'
id='roatateLeft'>"
 };
 object tools2 = new
 {
 tooltipText = "Rotate Right",

```

```

 template = "<button class='e-tbar-btn e-btn'
id='roataateRight'>"
 };
 ViewBag.image = new[] {
 "Replace", "Align", "Caption", "Remove", "InsertLink",
"OpenImageLink", "-",
 "EditImageLink", "RemoveImageLink", "Display", "AltText",
"Dimension", tools1
 , tools2
 };
 return View();
}
}

```

## See Also

- [How to render the toolbar in inline mode](#)

## Styling

### Font Name and Font Size

By default, the editor is initialized with **Segoe UI** font name and **10pt** font size. To change it, select a different font name and font size from the drop-down in the editor's toolbar.

To apply different font style for section of the content, select the text that you would like to change, and select a required font style from the drop-down to apply the changes to the selected text.

#### FontName DropDowns

The following table list the default font name and width of the [FontFamily](#) dropdown and available list of font names.

| Default Key | Default Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| -----       | -----                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| font name   | null                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| width       | 65px                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| items       | { text: 'Segoe UI', value: 'Segoe UI' }, { text: 'Arial', value: 'Arial,Helvetica,sans-serif' }, { text: 'Courier New', value: 'Courier New,Courier,monospace' }, { text: 'Georgia', value: 'Georgia,serif' }, { text: 'Impact', value: 'Impact,Charcoal,sans-serif' }, { text: 'Lucida Console', value: 'Lucida Console,Monaco,monospace' }, { text: 'Tahoma', value: 'Tahoma,Geneva,sans-serif' }, { text: 'Times New Roman', value: 'Times New Roman,Times,serif' }, { text: 'Trebuchet MS', value: 'Trebuchet MS,Helvetica,sans-serif' }, { text: 'Verdana', value: 'Verdana,Geneva,sans-serif' } |

#### FontSize DropDowns

The following table list the default font size and width of the [FontSize](#) dropdown and available list of font size.

| Default Key | Default Value |
|-------------|---------------|
| -----       | -----         |
| font size   | null          |



```
| width | 35px.|
```

```
| items [{ text: '8 pt', value: '8pt' }, { text: '10 pt', value: '10pt' }, { text: '12 pt', value: '12pt' }, { text: '14 pt', value: '14pt' }, { text: '18 pt', value: '18pt' }, { text: '24 pt', value: '24pt' }, { text: '36 pt', value: '36pt' }].|
```

The below sample demonstrate the option to add the font name and font size tools to the toolbar as well as modify the default **width** of the tools.

### CSHTML

```
@Html.EJS().RichTextEditor("font").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).FontFamily(t =>
t.Width("100px")).FontSize(t=>t.Width("40px")).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "FontName", "FontSize" };
 return View();
 }
}
```

### Custom Fonts and Size

Rich Text Editor supports to provide custom font and size with existing list.

If you want to add additional font names and font sizes to font drop-down, pass the font information as JSON data to the items field of [FontSize](#) and [FontFamily](#) property.

**CSHTML**

```
@Html.EJS().RichTextEditor("custom-font").ToolBarSettings(e =>
e.Items((object)ViewBag.toolbar)).FontFamily(f =>
f.Width("80px").default("Arial").Items((object)ViewBag.fontFamilyItems)).FontSize(f => f.Width("60px").default("10px")).Items((object)ViewBag.fontSizeItems)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads, etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Render()
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 object size1 = new
 {
 text = "8 px",
 value = "8px"
 };
 object size2 = new
 {
 text = "10 px",
 value = "10px"
 };
 object size3 = new
 {
 text = "12 px",
 value = "12px"
 };
 object size4 = new
 {
```

```

 text = "14 px",
 value = "14px"
 };
 object size5 = new
 {
 text = "42 px",
 value = "42px"
 };
 object item1 = new
 {
 text = "Segoe UI",
 value = "Segoe UI"
 };
 object item2 = new
 {
 text = "Arial",
 value = "Arial,Helvetica,sans-serif"
 };
 object item3 = new
 {
 text = "Courier New",
 value = "Courier New,Courier,monospace"
 };
 object item4 = new
 {
 text = "Georgia",
 value = "Georgia,serif"
 };
 object item5 = new
 {
 text = "Impact",
 value = "Impact,Charcoal,sans-serif"
 };
 object item6 = new
 {
 text = "Calibri Light",
 value = "CalibriLight"
 };
 ViewBag.items = new[] { "FontName", "FontSize" };
 ViewBag.fontFamilyItems = new[] { item1, item2, item3, item4, item5,
item6 };
 ViewBag.fontSizeItems = new[] { size1, size2, size3, size4, size5 };
 return View();
}
}

```

### Font and Background Color

If you want to apply font color or background color for a selected content of Rich Text Editor you can use font color and background color tools.

Rich Text Editor support to provide customs font color and background color with existing list through the `colorCode` field of [FontColor](#) and [BackgroundColor](#).

**FontColor** and **BackgroundColor** property has two mode **Picker** and **Palette**. Palette mode has predefined set of **colorCode** and in picker mode, we have provided with more colors. Through **modeSwitcher** we can able to switch between these two options.

### CSHTML

```
@Html.EJS().RichTextEditor("edit-color").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).FontColor(f =>
f.ModeSwitcher(true)).BackgroundColor(b =>
b.ModeSwitcher(true)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "FontColor", "BackgroundColor" };
 return View();
 }
}
```

### Editor content styles

By default, The content styles of Rich Text Editor are not returned while retrieving HTML value from the editor. So, the styles are not applied when using the HTML value outside of the editor. To get the styles to Rich Text Editor's content for your application, You can copy and use the below styles directly in your application. The styles listed below which used in the UI elements of the Rich Text Editor.

**Note:** Make sure to add a CSS class 'e-rte-content' to the content container.

```
`css
.e-rte-content p {
margin: 0 0 10px;
margin-bottom: 10px;
}
.e-rte-content li {
margin-bottom: 10px;
}
.e-rte-content h1 {
font-size: 2.17em;
font-weight: 400;
line-height: 1;
margin: 10px 0;
}
.e-rte-content h2 {
font-size: 1.74em;
font-weight: 400;
margin: 10px 0;
}
.e-rte-content h3 {
font-size: 1.31em;
font-weight: 400;
margin: 10px 0;
}
.e-rte-content h4 {
font-size: 1em;
font-weight: 400;
margin: 0;
}
.e-rte-content h5 {
font-size: 0.8em;
font-weight: 400;
margin: 0;
```

```
}
.e-rte-content h6 {
font-size: 00.65em;
font-weight: 400;
margin: 0;
}
.e-rte-content blockquote {
margin: 10px 0;
margin-left: 0;
padding-left: 5px;
}
.e-rte-content pre {
background-color: inherit;
border: 0;
border-radius: 0;
color: #333;
font-size: inherit;
line-height: inherit;
margin: 0 0 10px;
overflow: visible;
padding: 0;
white-space: pre-wrap;
word-break: inherit;
word-wrap: break-word;
}
.e-rte-content strong, .e-rte-content b {
font-weight: 700;
}
.e-rte-content a {
text-decoration: none;
-webkit-user-select: auto;
-ms-user-select: auto;
user-select: auto;
```

```
}
.e-rte-content a:hover {
text-decoration: underline;
}
.e-rte-content h3 + h4,
.e-rte-content h4 + h5,
.e-rte-content h5 + h6 {
margin-top: 00.6em;
}
.e-rte-content .e-rte-image.e-imgbreak {
border: 0;
cursor: pointer;
display: block;
float: none;
margin: 5px auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image {
border: 0;
cursor: pointer;
display: block;
float: none;
margin: auto;
max-width: 100%;
position: relative;
}
.e-rte-content .e-rte-image.e-imginline {
display: inline-block;
float: none;
margin-left: 5px;
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
```

```
vertical-align: bottom;
}

.e-rte-content .e-rte-image.e-imgcenter {
cursor: pointer;
display: block;
float: none;
margin: 5px auto;
max-width: 100%;
position: relative;
}

.e-rte-content .e-rte-image.e-imleft {
float: left;
margin: 0 5px 0 0;
text-align: left;
}

.e-rte-content .e-rte-image.e-imgright {
float: right;
margin: 0 0 0 5px;
text-align: right;
}

.e-rte-content .e-rte-img-caption {
display: inline-block;
margin: 5px auto;
max-width: 100%;
position: relative;
}

.e-rte-content .e-rte-img-caption.e-caption-inline {
display: inline-block;
margin: 5px auto;
margin-left: 5px;
margin-right: 5px;
max-width: calc(100% - (2 * 5px));
position: relative;
```



```
text-align: center;
vertical-align: bottom;
}
.e-rte-content .e-rte-img-caption.e-imgcenter {
display: block;
}
.e-rte-content .e-rte-img-caption .e-rte-image.e-imgright,
.e-rte-content .e-rte-img-caption .e-rte-image.e-imleft {
float: none;
margin: 0;
}
.e-rte-content .e-rte-table {
border-collapse: collapse;
empty-cells: show;
}
.e-rte-content .e-rte-table td,
.e-rte-content .e-rte-table th {
border: 1px solid #bdbdbd;
height: 20px;
min-width: 20px;
padding: 2px 5px;
vertical-align: middle;
}
.e-rte-content .e-rte-table.e-dashed-border td,
.e-rte-content .e-rte-table.e-dashed-border th {
border-style: dashed;
}
.e-rte-content .e-rte-img-caption .e-img-inner {
box-sizing: border-box;
display: block;
font-size: 16px;
font-weight: initial;
margin: auto;
```

```

opacity: .9;
position: relative;
text-align: center;
width: 100%;
}
.e-rte-content .e-rte-img-caption .e-img-wrap {
display: inline-block;
margin: auto;
padding: 0;
width: 100%;
}
.e-rte-content blockquote {
border-left: solid 2px #333;
}
.e-rte-content a {
color: #2e2ef1;
}
.e-rte-content .e-rte-table th {
background-color: #e0e0e0;
}
`

```

### See Also

- [How to add google fonts to the font family](#)
- [How to customize the placeholder](#)

### Image

Rich Text Editor allows to insert images from online sources as well as local computer where you want to insert the image in your content. For inserting the image to the Rich Text Editor, the following list of options have been provided in the [InsertImageSettings](#)

| Options | Description |

|-----|-----|

| AllowedTypes | Specifies the extensions of the image types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as .jpg and .png. |

| Display | Sets the default display for an image when it is inserted in to the Rich Text Editor. Possible options are: 'inline' and 'break'. |

- | Width | Sets the default width of the image when it is inserted in the Rich Text Editor.
- | Height | Sets the default height of the image when it is inserted in the Rich Text Editor.
- | SaveUrl | Provides URL to map the action result method to save the image.
- | Path | Specifies the location to store the image.
- | Resize | To enable resizing for image element.
- | MinWidth | Defines the maximum Width of the image.
- | MaxWidth | Defines the maximum Width of the image.
- | MinHeight | Defines the minimum Height of the image.
- | MaxHeight | Defines the maximum Height of the image.
- | ResizeByPercent | Image resizing should be done by percentage calculation.

### Upload Options

Through the **browse** option in the Image dialog, select the image from the local machine and insert into the Rich Text Editor content.

If the path field is not specified in the [InsertImageSettings](#), the image will be converted into blob url for the image will be created and the generated url will be set as src property of `<img>` tag as below.

`typescript

```

```

**Note:** If you want to insert a lot of tiny images in the editor and don't want a specific physical location for saving images, you can opt to save format as Base64.

In the below sample, the image has been load from the local machine and it will be saved in the given location.

### CSHTML

```
@Html.EJS().RichTextEditor("image").ToolBarSettings(e =>
e.Items((object)ViewBag.items)).InsertImageSettings(obj =>
obj.Display("inline")).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>

</div>
```

```

 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Image" };
 return View();
 }
}

```

## Server Side Action

The selected image can be uploaded to the required destination by using the below controller action. Map this method name in saveUrl property of [InsertImageSettings](#) and provide required destination path through path property.

**Note:** The runnable demo application is available in this [Github](#) repository.

## CSHTML

```

@Html.EJS().RichTextEditor("defaultRTE").InsertImageSettings(obj =>
obj.SaveUrl("/Home/SaveImage").Path("../Uploads/")).Render()

```

## SAVEFILE.CS

```

using System.IO;
using System.Web;
using System.Web.Mvc;
namespace ImageUpload.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveImage(HttpPostedFileBase UploadFiles)
 {
 if (UploadFiles != null)
 {
 string path = Server.MapPath("~/Uploads/");
 // Create a new directory, if it does not exists
 if (!Directory.Exists(path))
 {

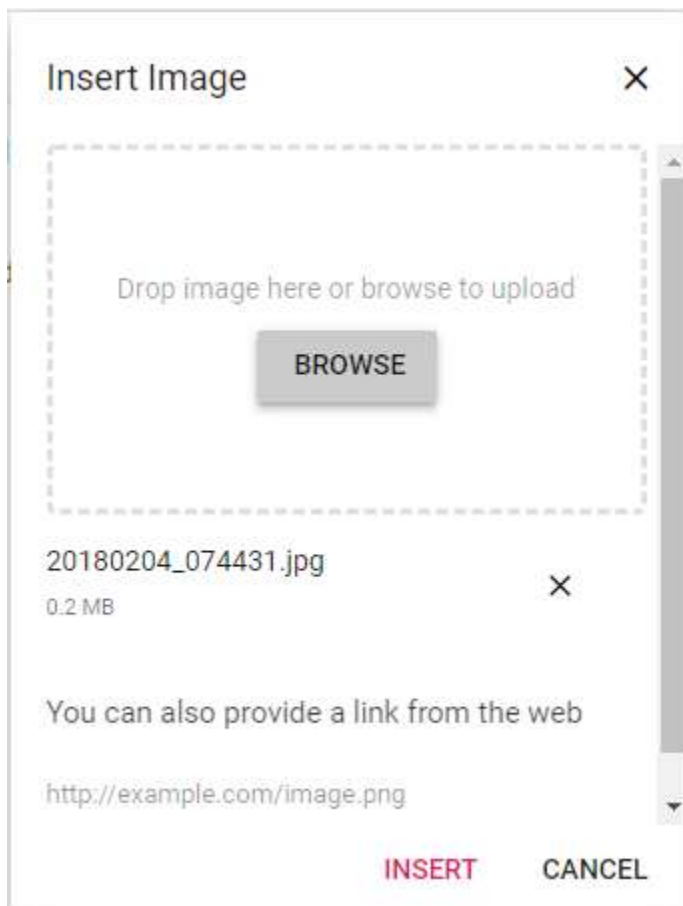
```

```
 Directory.CreateDirectory(path);
 }
 // Save a image file in directory
 UploadFiles.SaveAs(path +
Path.GetFileName(UploadFiles.FileName));
 }
}
}
```

### Image Delete

If you want to remove an image from the Rich Text Editor content, select the image and click “Remove” tool from quick toolbar. It will delete the image from the Rich Text Editor content.

Once you select the image from the local machine, the URL for the image will be generated, from there also you remove the image from the service location through the clicking of cross icon as below in the image.



The following sample explains, how to configure `RemoveUrl` to remove a saved image from the remote service location, when the following image remove actions are performed:

- `delete` key action.
- `backspace` key action.
- Removing uploaded image file from the insert image dialog.

- Deleting image using the quick toolbar **remove** option.

### CSHTML

```
@Html.EJS().RichTextEditor("RteImage").InsertImageSettings(obj =>
obj.SaveUrl("/Home/SaveImage").RemoveUrl("/Home/RemoveImage").Path("../Uploads/")).Render()
```

### HOME-CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveImage(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 string fileName = UploadFiles.FileName;
 var fileSave =
System.Web.HttpContext.Current.Server.MapPath("~/Uploads");
 // Create a new directory, if it does not exists
 if (!System.IO.Directory.Exists(fileSave))
 {
 System.IO.Directory.CreateDirectory(fileSave);
 }
 var fileSavePath = Path.Combine(fileSave, fileName);
 // Save the image
 UploadFiles.SaveAs(fileSavePath);
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 }
 }
 catch (Exception)
 {
 Response.StatusCode = 204;
 }
 }
 [AcceptVerbs("Post")]
 public void RemoveImage(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 // Do remove action here
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 }
 }
 }
}
```

```
 catch (Exception)
 {
 Response.StatusCode = 204;
 }
 }
```

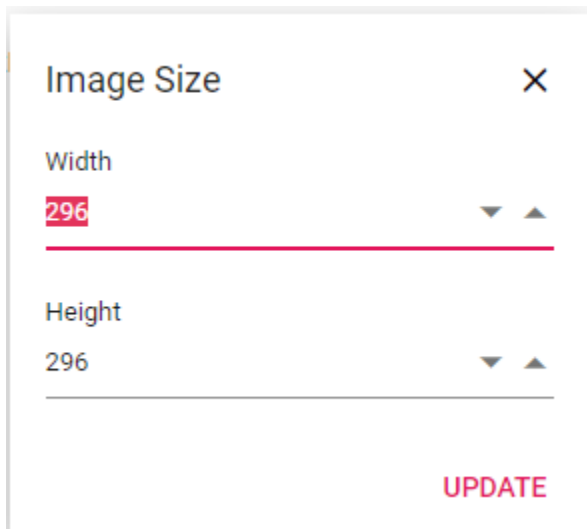
### Insert from Web

If you want to insert an image from online source like Google, Bing, etc., you need to enable images tool on the editor's toolbar. By default, the images tool is open an image dialog which allows you to insert an image from online source.

### Dimension

Sets the default width and height of the image when it is inserted in the Rich Text Editor using **Width** and **Height** of [InsertImageSettings](#).

Through the **quickToolbar**, also you can change the width and height using **Change Size** option. Once click into the option. The Image Size dialog will open as below. In that specify the width and height of the image in pixel.



### Caption and Alt Text

Image caption and alternative text can be specified for the inserted image in the Rich Text Editor through the [QuickToolbarSettings](#) options such as Image Caption and Alternative Text.

Through the Alternative Text option, set the alternative text for the image, when the image is not upload successfully into the Rich Text Editor.

By clicking the Image Caption, the image will get wrapped in an image element with a caption. Then, you can type caption content inside the Rich Text Editor.

### Display Position

Sets the default display for an image when it is inserted in the Rich Text Editor using **Display** field in [InsertImageSettings](#). It has two possible options: 'inline' and 'break'.

### CSHTML

```
@Html.EJS().RichTextEditor("image").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).InsertImageSettings(obj => obj.Display("break")).Render()
```

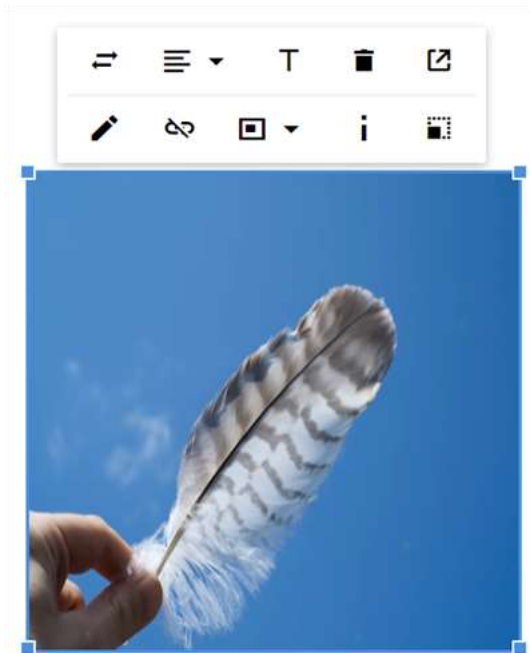
## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Image" };
 return View();
 }
}
```

### Image with Link

The hyperlink itself can be an image in Rich Text Editor. If the image given as hyperlink, the remove, edit and open link will be added to the quick toolbar of image as below. For further details about link, see the [link documentation](#).





### Drag and Drop

By default, the Rich Text Editor allows you to insert images by drag-and-drop from the local file system such as Windows Explorer into the content editor area. And, you can upload the images to the server before inserting into the editor by configuring the `saveUrl` property. The images can be repositioned anywhere within the editor area by dragging and dropping the image.

In the following sample, you can see feature demo.

### CSHTML

```
@Html.EJS().RichTextEditor("drag-drop").InsertImageSettings(obj =>
obj.SaveUrl("https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save"))
.ContentTemplate(@<div><p>
 The Rich Text Editor control is WYSIWYG ('what you see is what
 you get') editor that provides the best user experience to create and update
 the content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>

</div>
```

```
<p> Creates bulleted and numbered lists.</p>
</div>).Render()
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### *Drag and drop with specific extension images*

You can allow the specific images alone to be uploaded using the the `allowedTypes` property. By default, the Rich Text Editor allows the JPG, JPEG, and PNG formats. You can configure this formats as follows.

```
`typescript
```

```
insertImageSettings: {
 allowedTypes: ['.jpg']
},
```

### *Prevent drag and drop action*

You can prevent drag-and-drop action by setting the `actionBegin` argument cancel value to true. The following code shows how to prevent the drag-and-drop.

```
`typescript
```

```
function actionBegin(args) {
 if(args.type === 'drop' || args.type === 'dragstart') {
 args.cancel =true;
 }
}
```

### See Also

- [How to edit the quick toolbar settings](#)
- [How to use link editing option in the toolbar items](#)

## Insert Audio

The Rich Text Editor allows inserting audio files from online sources and the local computer where you want to insert the audio in your content. For inserting the audio to the Rich Text Editor, the following list of options have been provided in the [InsertAudioSettings](#).

| Options | Description |

|-----|-----|

| AllowedTypes | Specifies the extensions of the audio types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as `.mp3`, `.wav`, `.m4a` and `.wma` |

| LayoutOption | Sets the default display for audio when it is inserted into the Rich Text Editor. Possible options are: `Inline` and `Break`. |

| SaveFormat | Sets the default save format of the audio element when inserted. Possible options are: `Blob` and `Base64`. |

| SaveUrl | Provides URL to map the action result method to save the audio. |

| RemoveUrl | Provides URL to map the action result method to remove the audio. |

| Path | Specifies the location to store the audio. |

[Configure audio tool in the toolbar](#)

To include the audio tool in the Rich Text Editor, you can add the toolbar item `Audio` to the `ToolbarSettings` [Items](#) property.

To configure `Audio` toolbar item, refer to the below code.

#### **CSHTML**

```
@Html.EJS().RichTextEditor("editor").ToolbarSettings(e =>
 e.Items((object)ViewBag.items)).Render()
```

#### **CONTROLLER.CS**

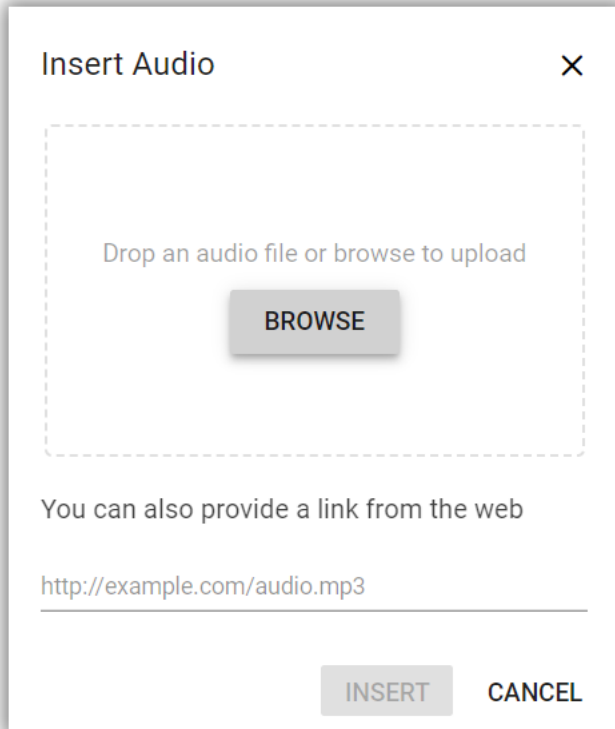
```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Audio" };
 return View();
 }
}
```

[Insert audio from web](#)

To insert audio from the hosted link or local machine, you should enable the audio tool on in the editor's toolbar.

[Insert from web URL](#)

By default, the audio tool opens the audio dialog, allowing you to insert audio from an online source. Inserting the URL will be added to the `src` attribute of the `<source>` tag.



### Upload and insert audio

In the audio dialog, using the **browse** option, select the audio from the local machine and insert it into the Rich Text Editor content.

If the path field is not specified in the [InsertAudioSettings](#), the audio will be converted into **Blob** url or **Base64** and inserted inside the Rich Text Editor.

### Restrict audio upload based on size

You can restrict the audio uploaded from the local machine when the uploaded audio file size is greater than the allowed size by using the [FileUploading](#) event.

**Note:** The file size in the argument will be returned in **bytes**.

Using the Rich Text Editor **FileUploading** event, you can restrict the audio to upload when the given audio size is greater than the allowed **fileSize**. Also, the audio size in the argument will be returned in **bytes**.

In the following illustration, the audio size has been validated before uploading, and it is determined whether the audio has been uploaded or not.

### CSHTML

```
@Html.EJS().RichTextEditor("editor").ToolbarSettings(e =>
e.Items((object) ViewBag.items)).InsertAudioSettings(obj =>
obj.SaveUrl("https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save").
Path("../Files/")).FileUploading("onFileUploading").Render()
<script>
 function onFileUploading(args) {
 var imgSize = 500000;
 var sizeInBytes = args.fileData.size;
```

```

 if (imgSize < sizeInBytes) {
 args.cancel = true;
 }
 }
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Audio" };
 return View();
 }
}

```

#### *Server-side action*

The selected audio can be uploaded to the required destination using the controller action below. Map this method name in [InsertAudioSettings.SaveUrl](#) and provide the required destination path through [InsertAudioSettings.Path](#) properties.

**Note:** If you want to insert lower-sized audio files in the editor and don't want a specific physical location for saving audio, you can opt to save the format as **Base64**.

In the following code blocks, you can insert the audio files which are saved in the specified path.

### CSHTML

```

@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).InsertAudioSettings(obj =>
obj.SaveUrl("/Home/SaveFiles").Path("../Files/")).Render()

```

### SAVEFILE.CS

```

using System.IO;
using System.Web;
using System.Web.Mvc;
namespace FileUpload.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Audio" };
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveFiles(HttpPostedFileBase UploadFiles)
 {
 if (UploadFiles != null)
 {
 string path = Server.MapPath("~/Files/");
 // Create a new directory, if it does not exists
 if (!Directory.Exists(path))

```

```

 {
 Directory.CreateDirectory(path);
 }
 // Save a file in directory
 UploadFiles.SaveAs(path +
Path.GetFileName(UploadFiles.FileName));
 }
}
}
}

```

### Audio save format

The audio files can be saved as **Blob** or **Base64** url by using the [InsertAudioSettings.SaveFormat](#) property, which is of enum type and the generated url will be set to the **src** attribute of the **<source>** tag.

**Note:** The default **SaveFormat** property is set to **Blob** format.

By default, the files are saved in the **Blob** format.

`typescript

<audio>

<source src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879"  
type="audio/mp3" >

</audio>

<audio>

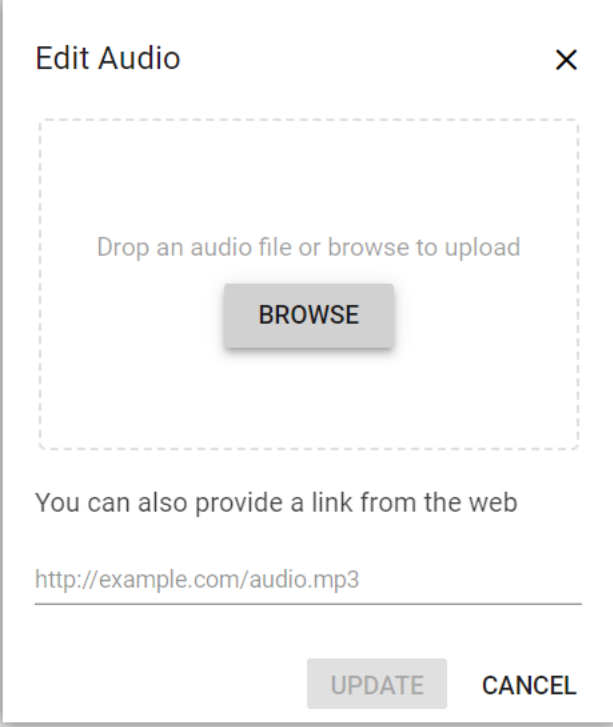
<source src="data:audio/mp3;base64,iVBORw0KGgoAAAANSUhEUgAAADAAAAAwCAYAAABXAvmHA"  
type="audio/mp3" >

</audio>

,

### Replacing audio

Once an audio file has been inserted, you can change it using the Rich Text Editor [QuickToolbarSettings](#) **audioReplace** option. You can replace the audio file using the web URL or the browse option in the audio dialog.

The image shows a modal dialog box titled "Edit Audio" with a close button (X) in the top right corner. Inside the dialog, there is a dashed rectangular area containing the text "Drop an audio file or browse to upload" and a "BROWSE" button. Below this area, the text "You can also provide a link from the web" is followed by a text input field containing the URL "http://example.com/audio.mp3". At the bottom of the dialog, there are two buttons: "UPDATE" and "CANCEL".

Edit Audio

Drop an audio file or browse to upload

BROWSE

You can also provide a link from the web

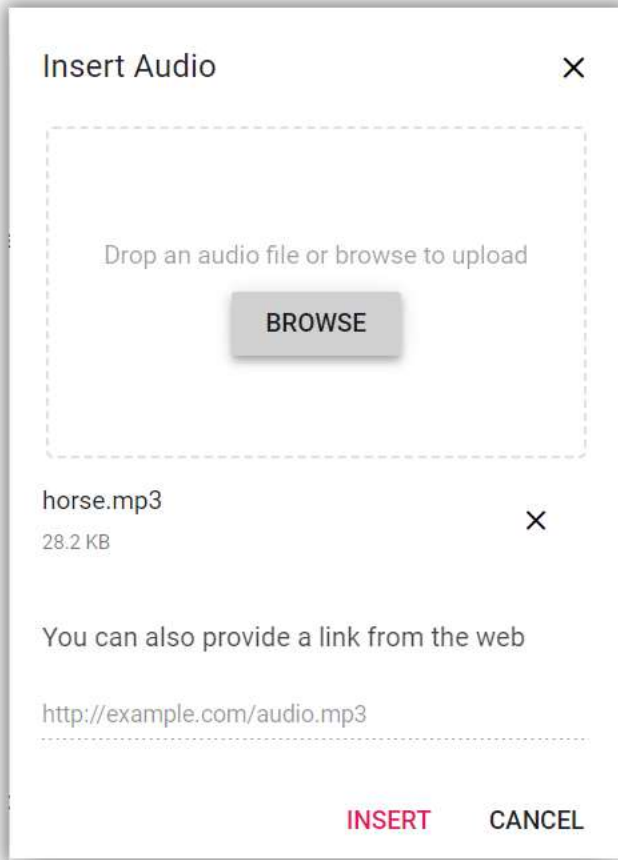
http://example.com/audio.mp3

UPDATE CANCEL

### Delete audio

To remove audio from the Rich Text Editor content, select the audio and click the `audioRemove` tool from the quick toolbar. It will delete the audio from the Rich Text Editor content as well as from the service location if the [InsertAudioSettings.RemoveUrl](#) is given.

Once you select the audio from the local machine, the URL for the audio will be generated. You can remove the audio from the service location by clicking the cross icon.



The following sample explains how to configure `InsertAudioSettings.RemoveUrl` to remove saved audio from the remote service location when the following audio removal actions are performed:

- `delete` key action.
- `backspace` key action.
- Removing uploaded audio file from the insert audio dialog.
- Deleting audio using the quick toolbar `audioRemove` option.

**Note:** The default `LayoutOption` property is set to `Inline`.

#### CSHTML

```
@Html.EJS().RichTextEditor("RteImage").InsertImageSettings(obj =>
obj.SaveUrl("/Home/SaveImage").RemoveUrl("/Home/RemoveImage").Path("../Uploads/")).Render()
```

#### HOME-CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```



```

[AcceptVerbs("Post")]
public void SaveImage(HttpPostedFileBase UploadFiles)
{
 try
 {
 if (UploadFiles != null)
 {
 string fileName = UploadFiles.FileName;
 var fileSave =
System.Web.HttpContext.Current.Server.MapPath("~/Uploads");
 // Create a new directory, if it does not exists
 if (!System.IO.Directory.Exists(fileSave))
 {
 System.IO.Directory.CreateDirectory(fileSave);
 }
 var fileSavePath = Path.Combine(fileSave, fileName);
 // Save the image
 UploadFiles.SaveAs(fileSavePath);
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 }
 }
 catch (Exception)
 {
 Response.StatusCode = 204;
 }
}

[AcceptVerbs("Post")]
public void RemoveImage(HttpPostedFileBase UploadFiles)
{
 try
 {
 if (UploadFiles != null)
 {
 // Do remove action here
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 }
 }
 catch (Exception)
 {
 Response.StatusCode = 204;
 }
}
}

```

### Display Position

Sets the default display for an audio when it is inserted in the Rich Text Editor using the `InsertAudioSettings.LayoutOption`. It has two possible options: `Inline` and `Break`. When updating the display positions, it updates the audio elements' layout position.

### CSHTML

```

@Html.EJS().RichTextEditor("audio").ToolBarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>

```

```

<p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
</p>
<p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
</p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).InsertAudioSettings(obj => obj.LayoutOption("Break")).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Audio" };
 return View();
 }
}

```

### Rename audio before inserting

Using the [InsertAudioSettings](#) property, you can specify the server handler to upload the selected audio. Then by binding the `FileUploadSuccess` event, you can receive the modified file name from the server and update it in the Rich Text Editor's insert audio dialog.

Refer `rename.cs` controller file for configure the server-side.

## CSHTML

```

@Html.EJS().RichTextEditor("editor").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).InsertAudioSettings(obj =>
obj.SaveUrl("/Home/Rename").Path("../Uploads/")).FileUploadSuccess("onFileUp
loadSuccess").Render()
<script>
 function onFileUploadSuccess(args) {
 if (args.e.currentTarget.getResponseHeader('name') != null) {
 args.file.name = args.e.currentTarget.getResponseHeader('name');
 var filename = document.querySelector(".e-file-name")[0];

```

```

 filename.innerHTML =
args.file.name.replace(document.querySelectorAll(".e-file-
type")[0].innerHTML, '');
 filename.title = args.file.name;
 }
}
</script>

```

## RENAME.CS

```

using System;
using System.IO;
using System.Web;
using System.Web.Mvc;
namespace RenameFile.Controllers
{
 public class HomeController : Controller
 {
 int x = 0;
 string filename;
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Audio" };
 return View();
 }
 [AcceptVerbs("Post")]
 public void Rename(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 filename = UploadFiles.FileName;
 if (UploadFiles != null)
 {
 var fileSave =
System.Web.HttpContext.Current.Server.MapPath("~/Uploads");
 // Create a new directory, if it does not exists
 if (!System.IO.Directory.Exists(fileSave))
 {
 System.IO.Directory.CreateDirectory(fileSave);
 }
 var fileName =
Path.GetFileName(UploadFiles.FileName);
 var fileSavePath = Path.Combine(fileSave, fileName);
 // Rename a uploaded file name
 while (System.IO.File.Exists(fileSavePath))
 {
 filename = "rteAudio" + x + "-" + fileName;
 fileSavePath = Path.Combine(fileSave, filename);
 x++;
 }
 if (!System.IO.File.Exists(fileSavePath))
 {
 UploadFiles.SaveAs(fileSavePath);

```

```

// Modified file name shared through response
header by adding custom header
 Response.Headers.Add("name", filename);
 Response.StatusCode = 200;
 Response.ContentType = "application/json;
charset=utf-8";
 }
}
}
}
}
catch (Exception)
{
 Response.StatusCode = 204;
}
}
}
}
}

```

### Upload audio with authentication

The Rich Text Editor control allows you to add additional data with the File Upload, which can be received on the server side. By using the `FileUploading` event and its `customFormData` argument, you can pass parameters to the controller action. On the server side, you can fetch the custom headers by accessing the form collection from the current request, which retrieves the values sent using the POST method.

**Note:** By default, it doesn't support the `UseDefaultCredentials` property; we need to manually append the default credentials with the upload request.

By default it doesn't support `UseDefaultCredentials` property, we need to manually append the default credentials with the upload request.

### CSHTML

```

@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object) ViewBag.items)).InsertAudioSettings(obj =>
obj.SaveUrl("/Home/SaveFiles").FileUploading("onFileUploading").Path("../Files/")).Render()
<script>
 function onFileUploading(args) {
 var accessToken = "Authorization_token";
 // adding custom Form Data
 args.customFormData = [{ 'Authorization': accessToken }];
 }
</script>

```

### SAVEFILE.CS

```

using System.IO;
using System.Web;
using System.Web.Mvc;
namespace FileUpload.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()

```

```

 {
 ViewBag.items = new[] { "Audio" };
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveFiles(HttpPostedFileBase UploadFiles)
 {
 string currentPath = Request.Form["Authorization"].ToString();
 if (UploadFiles != null)
 {
 string path = Server.MapPath("~/Files/");
 // Create a new directory, if it does not exists
 if (!Directory.Exists(path))
 {
 Directory.CreateDirectory(path);
 }
 // Save a file in directory
 UploadFiles.SaveAs(path +
 Path.GetFileName(UploadFiles.FileName));
 }
 }
}

```

### See Also

- [How to edit the quick toolbar settings](#)
- [How to use link editing option in the toolbar items](#)

### Insert Video

The Rich Text Editor allows you to insert videos from online sources and local computers where you want to insert the video in your content. For inserting the video to the Rich Text Editor, the following list of options have been provided in the [InsertVideoSettings](#).

| Options | Description |

|-----|-----|

| AllowedTypes | Specifies the extensions of the video types allowed to insert on bowering and passing the extensions with comma separators. For example, pass allowedTypes as `.mp4`, `.mov`, `.wmv` and `.avi`.|

| LayoutOption | Sets the default display for a video when it is inserted into the Rich Text Editor. Possible options are: `Inline` and `Break`.|

| SaveFormat | Sets the default save format of the video element when inserted. Possible options are: `Blob` and `Base64`.|

| Width | Sets the default width of the video when it is inserted in the Rich Text Editor.|

| MinWidth | Sets the minWidth of the video element when it is inserted in the Rich Text Editor.|

| MaxWidth | Sets the maxWidth of the video element when it is inserted in the Rich Text Editor.|

| Height | Sets the default height of the video when it is inserted in the Rich Text Editor.|

- | MinHeight | Sets the minHeight of the video element when it is inserted in the Rich Text Editor.
- | MaxHeight | Sets the maxHeight of the video element when it is inserted in the Rich Text Editor.
- | SaveUrl | Provides URL to map the action result method to save the video.
- | RemoveUrl | Provides URL to map the action result method to remove the video.
- | Path | Specifies the location to store the video.
- | Resize | Sets the resizing action for the video element.
- | ResizeByPercent | Sets the percentage values for the video element with the resizing action.

#### Configure video tool in the toolbar

To include the video tool in the Rich Text Editor, you can add the toolbar item **Video** to the **ToolbarSettings** [Items](#) property.

To configure **Video** toolbar item, refer to the below code.

#### CSHTML

```
@Html.EJS().RichTextEditor("editor").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).Render()
```

#### CONTROLLER.CS

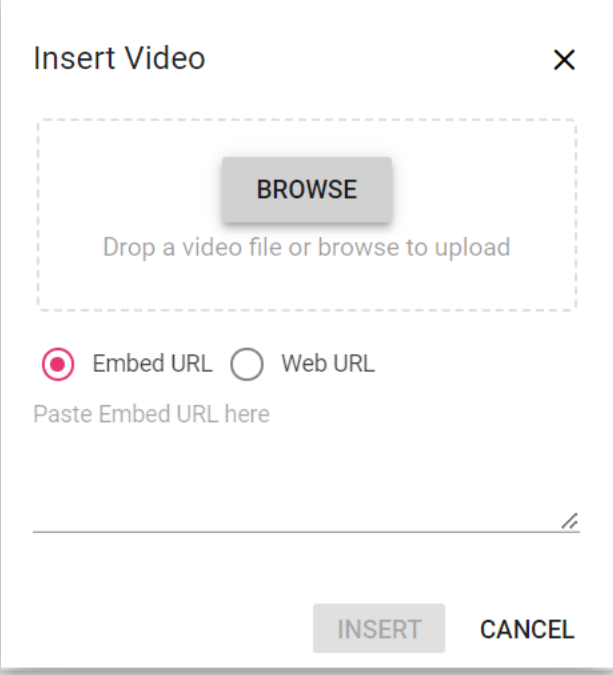
```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Video" };
 return View();
 }
}
```

#### Insert video from web

To insert a video from the hosted link or local machine, you should enable the video tool on the editor's toolbar. By default, the video tool opens the dialog, allowing you to insert a video as an embedded URL. You can switch to a web URL to insert the video file from the online source.

#### Insert from web URL

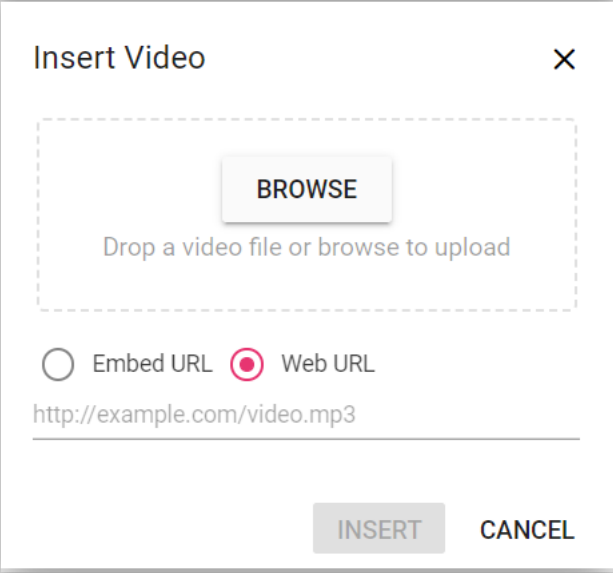
The video tool default opens the video dialog, allowing you to insert an embedded URL.



The dialog box is titled "Insert Video" with a close button (X) in the top right corner. It features a dashed rectangular area containing a "BROWSE" button and the text "Drop a video file or browse to upload". Below this, there are two radio buttons: "Embed URL" (which is selected with a red dot) and "Web URL". Under the "Embed URL" option, there is a text input field with the placeholder text "Paste Embed URL here". At the bottom of the dialog, there are two buttons: "INSERT" and "CANCEL".

#### *Insert from web URL*

Switching the option to the web URL in the video dialog allows you to insert a video from the online source. Inserting the URL will be added to the `src` attribute of the `<source>` tag.



The dialog box is titled "Insert Video" with a close button (X) in the top right corner. It features a dashed rectangular area containing a "BROWSE" button and the text "Drop a video file or browse to upload". Below this, there are two radio buttons: "Embed URL" and "Web URL" (which is selected with a red dot). Under the "Web URL" option, there is a text input field containing the example URL "http://example.com/video.mp3". At the bottom of the dialog, there are two buttons: "INSERT" and "CANCEL".

#### *Upload and insert video*

In the video dialog, by using the `browse` option, select the video from the local machine and insert it into the Rich Text Editor content.

If the path field is not specified in the [InsertVideoSettings](#), the video will be converted into `Blob` url or `Base64` and inserted inside the Rich Text Editor.

*Restrict video upload based on size*

Using the Rich Text Editor `FileUploading` event, you can restrict the video to upload when the given video size is greater than the allowed `fileSize`. Also, the video size in the argument will be returned in `bytes`.

**Note:** The file size in the argument will be returned in `bytes`.

In the following example, the video size has been validated before uploading and determined whether the video has been uploaded or not

In the following example, the video size has been validated before uploading and determined whether the video has been uploaded or not.

**CSHTML**

```
@Html.EJS().RichTextEditor("editor").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).InsertVideoSettings(obj =>
obj.SaveUrl("https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save").
Path("../Files/")).FileUploading("onFileUploading").Render()
<script>
 function onFileUploading(args) {
 var imgSize = 500000;
 var sizeInBytes = args.fileData.size;
 if (imgSize < sizeInBytes) {
 args.cancel = true;
 }
 }
</script>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Video" };
 return View();
 }
}
```

*Server-side action*

The selected video can be uploaded to the required destination using the controller action below. Map this method name in [InsertVideoSettings.SaveUrl](#) and provide required destination path through [InsertVideoSettings.Path](#) properties.

**Note:** If you want to insert lower-sized video files in the editor and don't want a specific physical location for saving the video, you can save the format as `Base64`.

In the following code blocks, you can insert the video files which are saved in the specified path.

**CSHTML**

```
@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).InsertVideoSettings(obj =>
obj.SaveUrl("/Home/SaveFiles").Path("../Files/")).Render()
```



**SAVEFILE.CS**

```

using System.IO;
using System.Web;
using System.Web.Mvc;
namespace FileUpload.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Video" };
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveFiles(HttpPostedFileBase UploadFiles)
 {
 if (UploadFiles != null)
 {
 string path = Server.MapPath("~/Files/");
 // Create a new directory, if it does not exists
 if (!Directory.Exists(path))
 {
 Directory.CreateDirectory(path);
 }
 // Save a file in directory
 UploadFiles.SaveAs(path +
 Path.GetFileName(UploadFiles.FileName));
 }
 }
 }
}

```

*Video save format*

The video files can be saved as **Blob** or **Base64** url by using the [InsertVideoSettings.SaveFormat](#) property, which is of enum type and the generated url will be set to the **src** attribute of the **<source>** tag.

**Note:** The default **SaveFormat** property is set to **Blob** format.

By default, the files are saved in the **Blob** format.

`typescript

<video>

<source src="blob:http://ej2.syncfusion.com/3ab56a6e-ec0d-490f-85a5-f0aeb0ad8879"  
type="video/mp4" >

</video>

<video>

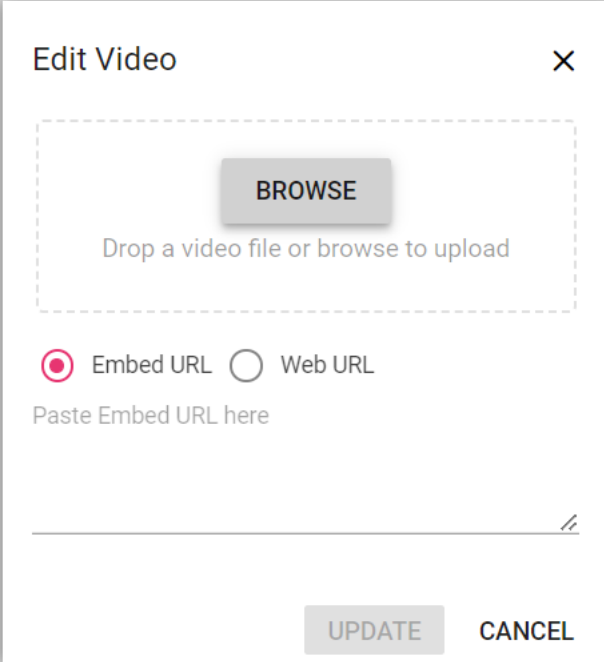
<source src="data:video/mp4;base64,iVBORw0KGgoAAAANSUHEUgAAADAAAAAwCAYAAABXAvmHA"  
type="video/mp4" >

&lt;/video&gt;

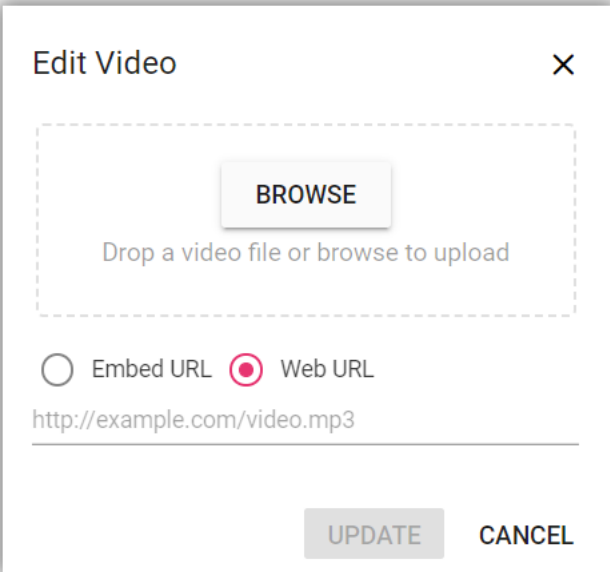
,

### Replacing video

Once a video file has been inserted, you can replace it using the Rich Text Editor [QuickToolbarSettings](#) `videoReplace` option. You can replace the video file either by using the embedded URL or the web URL and the browse option in the video dialog.



The 'Edit Video' dialog box features a close button (X) in the top right corner. It contains a dashed rectangular area with a 'BROWSE' button and the text 'Drop a video file or browse to upload'. Below this, there are two radio buttons: 'Embed URL' (which is selected) and 'Web URL'. A text input field labeled 'Paste Embed URL here' is positioned below the radio buttons. At the bottom, there are 'UPDATE' and 'CANCEL' buttons.

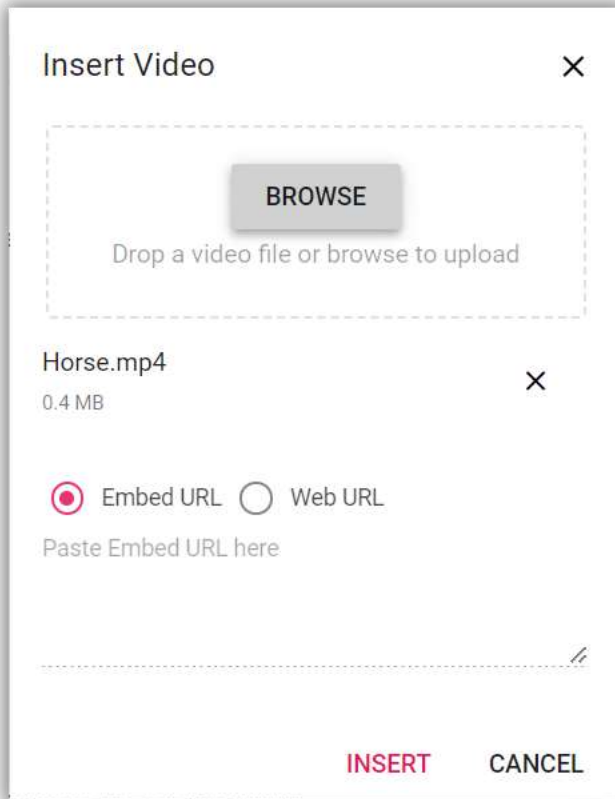


The 'Edit Video' dialog box is identical to the one above, but with the 'Web URL' radio button selected. The text input field now contains the example URL 'http://example.com/video.mp3'. The 'UPDATE' and 'CANCEL' buttons remain at the bottom.

### Delete video

To remove a video from the Rich Text Editor content, select the video and click the `videoRemove` tool from the quick toolbar. It will delete the video from the Rich Text Editor content as well as from the service location if the [InsertVideoSettings.RemoveUrl](#) is given.

Once you select the video from the local machine, the URL for the video will be generated. You can remove the video from the service location by clicking the cross icon.



The following example explains how to configure `InsertVideoSettings.removeUrl` to remove a saved video from the remote service location when the following video remove actions are performed:

- `delete` key action.
- `backspace` key action.
- Removing uploaded video file from the insert video dialog.
- Deleting video using the quick toolbar `videoRemove` option.

### CSHTML

```
@Html.EJS().RichTextEditor("RteImage").InsertImageSettings(obj =>
obj.SaveUrl("/Home/SaveImage").RemoveUrl("/Home/RemoveImage").Path("../Uploads/")).Render()
```

### HOME-CONTROLLER.CS

```
public class HomeController : Controller
```

```

{
 public ActionResult Index()
 {
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveImage(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 string fileName = UploadFiles.FileName;
 var fileSave =
System.Web.HttpContext.Current.Server.MapPath("~/Uploads");
 // Create a new directory, if it does not exists
 if (!System.IO.Directory.Exists(fileSave))
 {
 System.IO.Directory.CreateDirectory(fileSave);
 }
 var fileSavePath = Path.Combine(fileSave, fileName);
 // Save the image
 UploadFiles.SaveAs(fileSavePath);
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 }
 }
 catch (Exception)
 {
 Response.StatusCode = 204;
 }
 }
 [AcceptVerbs("Post")]
 public void RemoveImage(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 // Do remove action here
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 }
 }
 catch (Exception)
 {
 Response.StatusCode = 204;
 }
 }
}

```

### Dimension

Set the default width, minWidth, height and minHeight of the video element, when it is inserted in the Rich Text Editor using the [Width](#), [MinWidth](#), [Height](#), [MinHeight](#) properties.

Through the [QuickToolbarSettings](#), also you can change the width and height using **Change Size** option. Once you click on the option, the video size dialog will open as below. In that, specify the width and height of the video in pixels

The image shows a 'Video Size' dialog box with a close button (X) in the top right corner. It contains two input fields: 'Width' with the value '321px' and 'Height' with the value '241px'. A red 'UPDATE' button is located at the bottom right of the dialog.

### Display Position

Sets the default display property for the video when it is inserted in the Rich Text Editor using the `InsertVideoSettings.LayoutOption`. It has two possible options: **Inline** and **Break**. When updating the display positions, it updates the video elements' layout position.

**Note:** The default `LayoutOption` property is set to **Inline**.

Sets the default display for an video when it is inserted in the Rich Text Editor using the `InsertVideoSettings.LayoutOption`. It has two possible options: **Inline** and **Break**. When updating the display positions, it updates the video elements' layout position.

### CSHTML

```
@Html.EJS().RichTextEditor("video").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>

</div>
```

```

<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).InsertVideoSettings(obj => obj.LayoutOption("Break")).Render()

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Video" };
 return View();
 }
}

```

### Resize video

The Rich Text Editor has built-in video resizing support, which is enabled for the video elements added. The resize points will appear on each corner of the video when focusing so users can easily resize the video using mouse points or thumb through the resize points. Also, the resize calculation will be done based on the aspect ratio.

You can disable the resize action by configuring `false` for the [InsertVideoSettings.Resize](#) property.

**Note:** If the [MinWidth](#) and [MinHeight](#) properties are configured the video resizing does not shrink below the specified values.



### Rename video before inserting

By using the [InsertVideoSettings](#) property, you can specify the server handler to upload the selected video. Then by binding the `FileUploadSuccess` event, you can receive the modified file name from the server and update it in the Rich Text Editor's insert video dialog.

Refer `rename.cs` controller file for configure the server-side.

### CSHTML

```
@Html.EJS().RichTextEditor("editor").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).InsertVideoSettings(obj =>
obj.SaveUrl("/Home/Rename").Path("../Uploads/")).FileUploadSuccess("onFileUp
loadSuccess").Render()
<script>
 function onFileUploadSuccess(args) {
 if (args.e.currentTarget.getResponseHeader('name') != null) {
 args.file.name = args.e.currentTarget.getResponseHeader('name');
 var filename = document.querySelectorAll(".e-file-name")[0];
 filename.innerHTML =
args.file.name.replace(document.querySelectorAll(".e-file-
type")[0].innerHTML, '');
 filename.title = args.file.name;
 }
 }
</script>
```

**RENAME.CS**

```
using System;
using System.IO;
using System.Web;
using System.Web.Mvc;
namespace RenameFile.Controllers
{
 public class HomeController : Controller
 {
 int x = 0;
 string filename;
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Video" };
 return View();
 }
 [AcceptVerbs("Post")]
 public void Rename(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 filename = UploadFiles.FileName;
 if (UploadFiles != null)
 {
 var fileSave =
System.Web.HttpContext.Current.Server.MapPath("~/Uploads");
 // Create a new directory, if it does not exists
 if (!System.IO.Directory.Exists(fileSave))
 {
 System.IO.Directory.CreateDirectory(fileSave);
 }
 var fileName =
Path.GetFileName(UploadFiles.FileName);
 var fileSavePath = Path.Combine(fileSave, fileName);
 // Rename a uploaded file name
 while (System.IO.File.Exists(fileSavePath))
```

```

 {
 filename = "rteVideo" + x + "-" + fileName;
 fileSavePath = Path.Combine(fileSave, filename);
 x++;
 }
 if (!System.IO.File.Exists(fileSavePath))
 {
 UploadFiles.SaveAs(fileSavePath);
 // Modified file name shared through response
 header by adding custom header
 Response.Headers.Add("name", filename);
 Response.StatusCode = 200;
 Response.ContentType = "application/json;
charset=utf-8";
 }
 }
}
catch (Exception)
{
 Response.StatusCode = 204;
}
}
}
}

```

## Upload video with authentication

The Rich Text Editor control allows you to add additional data with the File Upload, which can be received on the server side. By using the `FileUploading` event and its `customFormData` argument, you can pass parameters to the controller action. On the server side, you can fetch the custom headers by accessing the form collection from the current request, which retrieves the values sent using the POST method.

**Note:** By default, it doesn't support the `UseDefaultCredentials` property, you can manually append the default credentials with the upload request.

By default it doesn't support `UseDefaultCredentials` property, we need to manually append the default credentials with the upload request.

## CSHTML

```
@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).InsertVideoSettings(obj =>
obj.SaveUrl("/Home/SaveFiles").FileUploading("onFileUploading").Path("../Files/")).Render()

<script>
 function onFileUploading(args) {
 var accessToken = "Authorization_token";
 // adding custom Form Data
 args.customFormData = [{ 'Authorization': accessToken }];
 }
</script>
```

**SAVEFILE.CS**



```

using System.IO;
using System.Web;
using System.Web.Mvc;
namespace FileUpload.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Video" };
 return View();
 }
 [AcceptVerbs("Post")]
 public void SaveFiles(HttpPostedFileBase UploadFiles)
 {
 string currentPath = Request.Form["Authorization"].ToString();
 if (UploadFiles != null)
 {
 string path = Server.MapPath("~/Files/");
 // Create a new directory, if it does not exists
 if (!Directory.Exists(path))
 {
 Directory.CreateDirectory(path);
 }
 // Save a file in directory
 UploadFiles.SaveAs(path +
 Path.GetFileName(UploadFiles.FileName));
 }
 }
 }
}

```

## See Also

- [How to edit the quick toolbar settings](#)
- [How to use link editing option in the toolbar items](#)

## Link

A hyperlink can be insert into the editor for quick access to the related information. The hyperlink itself can be a text or an image.

### Insert Link

Point the cursor anywhere within the editor where you would like to insert the link. It is also possible to select a text or an image within the editor and can be converted to the hyperlink. Click the Insert HyperLink tool on the toolbar. The Insert Link Dialog will be open. The dialog has the following options.

| Options      | Description                                                               |
|--------------|---------------------------------------------------------------------------|
| ----- -----  |                                                                           |
| Web Address  | Type or paste the destination for the link you are creating               |
| Display Text | Type or edit the required text that you want to display text for the link |

| Tooltip | To display additional helpful information when you place the pointer on the hyperlink, type the required text in the “Tooltip” field. |

| Open Link in New Window | Specify whether, the given link will be open in new window or not |

### CSHTML

```
@Html.EJS().RichTextEditor("link").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "CreateLink" };
 return View();
 }
}
```

**Note:** The Rich Text Editor link tool validates URLs, as you type them in Web Address. URLs considered invalid will be highlighted with red color by clicking the insert button in the **Insert Link** dialog.

#### Remove Link

If you want to remove a hyperlink from a text or image, select the text or image with the hyperlink and click **Remove Hyperlink** tool from toolbar. It will keep the text or image.

### Auto-link

When you type URL and Enter key to the Rich Text Editor, the typed URL will be automatically changed into the hyperlink.

### Manipulation

It is possible to add custom style on the selected link inside the Rich Text Editor through quick toolbar.

The quick toolbar for the Link has the following options.

| Tools       | Description                                               |
|-------------|-----------------------------------------------------------|
| ----- ----- |                                                           |
| Open        | The given link page, will be open in new window.          |
| Edit Link   | Used to edit the link in the Rich Text Editor content.    |
| Remove Link | Used to remove link from the content of Rich Text Editor. |
| Custom Tool | Used to add the custom options in the quick toolbar.      |

### CSHTML

```
@Html.EJS().RichTextEditor("link").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "CreateLink" };
 }
}
```

```

 return View();
 }
}

```

See Also

- [How to edit the quick toolbar settings](#)
- [How to insert image link editing option in the toolbar items](#)

## Table

Rich Text Editor allows to insert table of content in edit panel and provide options to add, edit, and remove the table as well as perform other table related action. For inserting the table to the Rich Text Editor, the following list of options have been provided in the [TableSettings](#)

| Options  | Description                                                                                                                                                                                                                        | Default Value                   |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|
| minWidth | Sets the default minWidth of the table.                                                                                                                                                                                            | 0                               |
| maxWidth | Sets the default maxWidth of the table.                                                                                                                                                                                            | null                            |
| resize   | Enable resize feature in table.                                                                                                                                                                                                    | true                            |
| styles   | This is an array of key value pair, on each pair, key should be name of styling and value is class name. this list will be shown on quick toolbar options to change the styles of table on designing like dashed, double bordered. | <a href="#">TableStyleItems</a> |
| width    | Sets the default width of the table.                                                                                                                                                                                               | 100%                            |

## Insert Table

Using the [table](#) toolbar option, select a number of rows and columns to be inserted over the table grid and insert table into Rich Text Editor content using the mouse.

Tables can also be inserted through the [Insert Table](#) option in the pop-up where the number of rows and columns can be provided manually, and this is the default way in devices.

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "CreateTable" };
 return View();
 }
}

```

## Quick Toolbar

Quick toolbar is opened by clicking the table. It has different sets of commands to be performed on the table which increases the feasibility to edit the table easily.

### Table Header

**Table Header** command is available with quick toolbar option through which the header row can be added or removed from the inserted table. The following image illustrates the table header.



### Insert Rows

**Rows** can be inserted above or below the required table cell through the quick toolbar. Also, focused row can be deleted. The following screenshot shows the available options of the row item.



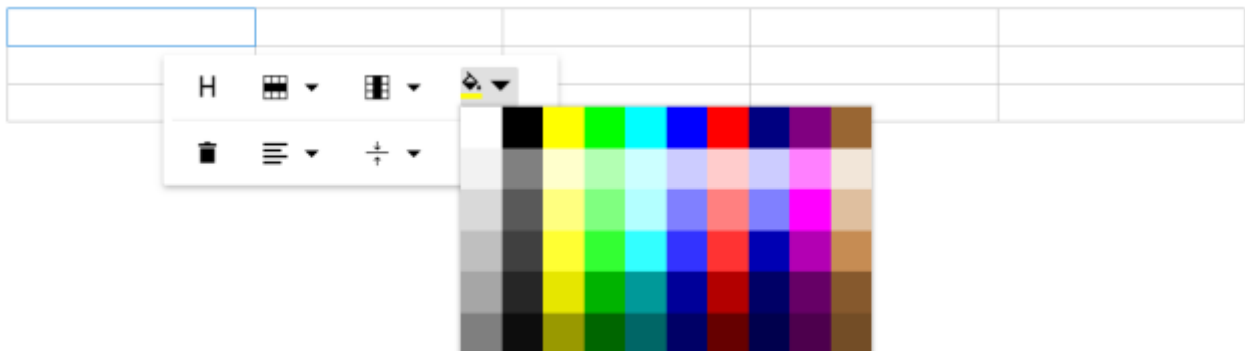
### Insert Columns

**Columns** can be inserted to the left or right side of the required table cell through the quick toolbar. Also, the focused column can be deleted. The following screenshot shows the available options of the column item.



### Set Color

The background color can be set for each table cell through the **background color** command available with quick toolbar.



### Delete Table

Using the delete item in the quick toolbar, users can delete the entire table.

### Vertical Align

Text inside the table can be aligned to top, middle, or bottom using the `tableCellVerticalAlign` tool of the quick toolbar.



### Horizontal Align

Text inside the table can be aligned left, right, or center using the `tableCellHorizontalAlign` tool of the quick toolbar.



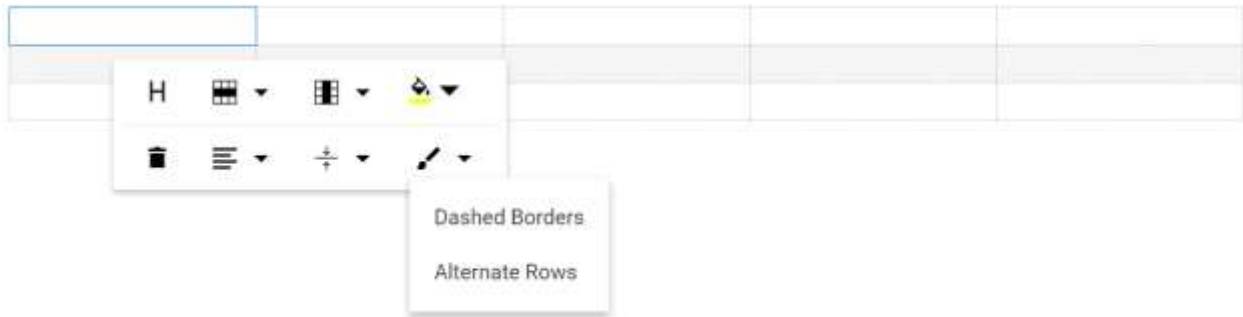
### Table Styles

Table styles provided for class name should be appended to a table element. It helps to design the table in specific CSS styles when inserting in the editor.

By Default, provides Dashed border and Alternate rows.

**Dashed border:** Applies the dashed border to the table.

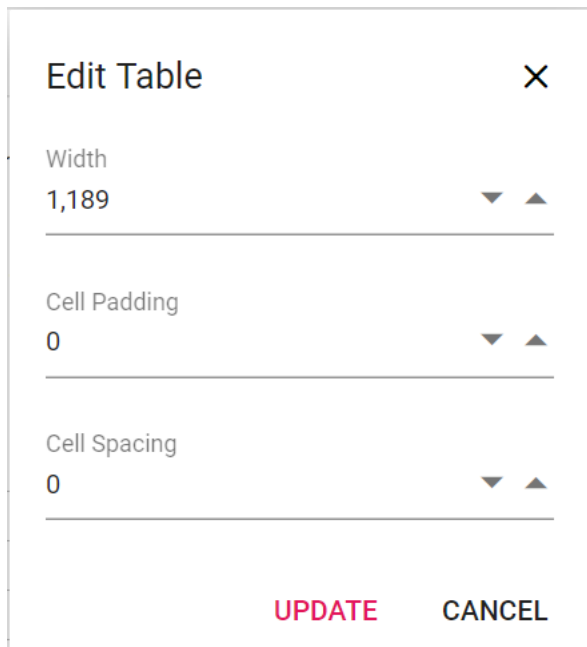
**Alternate border:** Applies the alternative background to the table.



### Table Properties

Sets the default width of the table when it is inserted in the Rich Text Editor using the width of [TableSettings](#).

Using the quick toolbar, users can change the width, cell padding, and cell spacing in the selected table using the properties option.



### Table cell merge and split

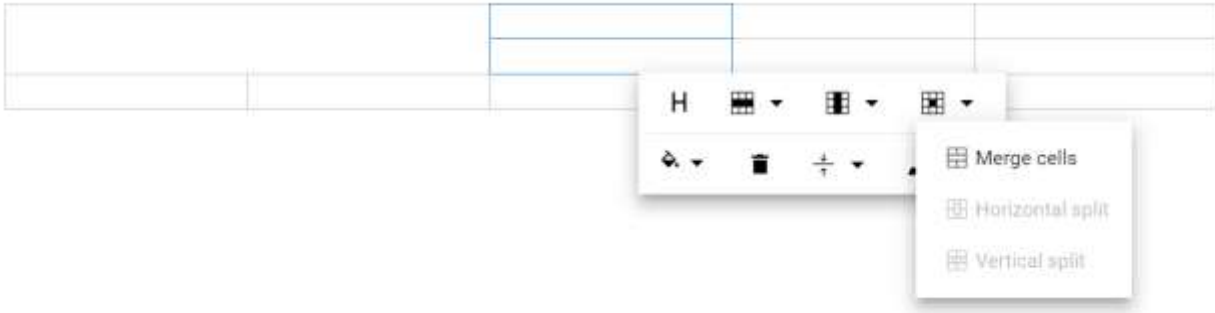
The Rich Text Editor allows users to change the appearance of the tables by splitting or merging the table cells.

**TableCell** item should be configured in the Table [quickToolbarSettings](#) Property to show the merge/split icons while selecting the table cells

#### Table cell merge

The table cell merge feature allows you to merge two or more row and column cells into a single cell with its contents.

The following image explains the table merge action.



### Table cell split

The table cell split feature allows you to split the merged cells both horizontally and vertically.

The following image explains the table split action.



## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "CreateTable" };
 ViewBag.Table = new[] {
 "TableHeader", "TableRows", "TableColumns", "TableCell", "-",
 "BackgroundColor", "TableRemove", "TableCellVerticalAlign", "Styles"
 };
 return View();
 }
}
```

### Emoji Picker

An emoji picker is a tool that allows users to add emojis or emoticons to their text easily. Typically, it is a small window or panel that displays a variety of emojis arranged in different categories, such as smileys, animals, food, and so on. Users can select the desired emoji by clicking on it or by typing its name in a search bar.

Enabling the toolbar option and custom emojis.

Add the **EmojiPicker** tool to the toolbar of the RichTextEditor by utilizing the **ToolbarSettings** [Items](#) property.

By default, a predefined set of emojis is configured. However, you can customize these icons according to your needs. To achieve this, utilize the [EmojiPickerSettings](#) property.

The following code example shows how to customize icons in the emoji picker.



**CSHTML**

```
<ejs-richtexteditor id="emojiPickerRTE">
 <e-richtexteditor-toolbarsettings items="ViewBag.items"/>
 <e-richtexteditor-emojipickersettings iconsSet= "ViewBag.iconsSet"/>
 <e-content-template>
 <p>An emoji picker in a Rich Text Editor is a tool that allows users
to easily add emojis or emoticons to their text.</p>
 <p>Typically, it is a small window or panel that displays a variety
of emojis, arranged in different categories, such as smileys, animals, food,
and so on. Users can select the desired emoji by clicking on it or by typing
its name in a search bar.</p>
 </e-content-template>
</ejs-richtexteditor>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[]
 {
 "EmojiPicker", "ClearFormat", "Bold", "Italic", "Underline",
 "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList", "|",
 "CreateLink", "Image", "|", "SourceCode", "Undo", "Redo"
 };
 ViewBag.iconsSet = new[]
 {
 new
 {
 name = "Smilies & People",
 code = "1F600",
 iconCss = "e-emoji",
 icons = new[]
 {
 new { code = "1F600", desc = "Grinning face" },
 new { code = "1F603", desc = "Grinning face with big
eyes" },
 new { code = "1F604", desc = "Grinning face with smiling
eyes" },
 new { code = "1F606", desc = "Grinning squinting face" },
 new { code = "1F605", desc = "Grinning face with sweat" },
 new { code = "1F602", desc = "Face with tears of joy" },
 new { code = "1F923", desc = "Rolling on the floor
laughing" },
 new { code = "1F60A", desc = "Smiling face with smiling
eyes" }
 }
 },
 new {
 name = "Animals & Nature",
 code = "1F435",
```

```

 iconCss = "e-animals",
 icons = new[] {
 new { code = "1F436", desc = "Dog face" },
 new { code = "1F431", desc = "Cat face" },
 new { code = "1F42D", desc = "Mouse face" },
 new { code = "1F439", desc = "Hamster face" },
 new { code = "1F430", desc = "Rabbit face" },
 new { code = "1F98A", desc = "Fox face" }
 },
 new {
 name = "Food & Drink",
 code = "1F347",
 iconCss = "e-food-and-drinks",
 icons = new[] {
 new { code = "1F34E", desc = "Red apple" },
 new { code = "1F34C", desc = "Banana" },
 new { code = "1F347", desc = "Grapes" },
 new { code = "1F353", desc = "Strawberry" },
 new { code = "1F35E", desc = "Bread" },
 new { code = "1F950", desc = "Croissant" },
 new { code = "1F955", desc = "Carrot" },
 new { code = "1F354", desc = "Hamburger" }
 },
 },
 new {
 name = "Activities",
 code = "1F383",
 iconCss = "e-activities",
 icons = new[] {
 new { code = "26BD", desc = "Soccer ball" },
 new { code = "1F3C0", desc = "Basketball" },
 new { code = "1F3C8", desc = "American football" },
 new { code = "26BE", desc = "Baseball" },
 new { code = "1F3BE", desc = "Tennis" },
 new { code = "1F3D0", desc = "Volleyball" },
 new { code = "1F3C9", desc = "Rugby football" }
 },
 },
 };
 return View();
}
}

```

Additionally, you have the option to customize the icons of toolbar items using the `IconCSS` and `Code` properties. The `IconCSS` property allows you to define a custom CSS class for the toolbar item icon, while the `Code` property enables you to specify the Unicode character code for the icon.

When both `IconCSS` and `Code` properties are provided, the `IconCSS` property takes precedence in determining the appearance of the toolbar item icon.

Additionally, you have the option to enhance the user experience by implementing a filtering feature for efficiently managing a large dataset of emojis. By setting the `ShowSearchBox` property to true (which is the default value), users will be able to utilize a search box to filter the displayed emojis according to their preferences.

The following code example shows how to add the emoji picker tool in the RichTextEditor.

### CSHTML

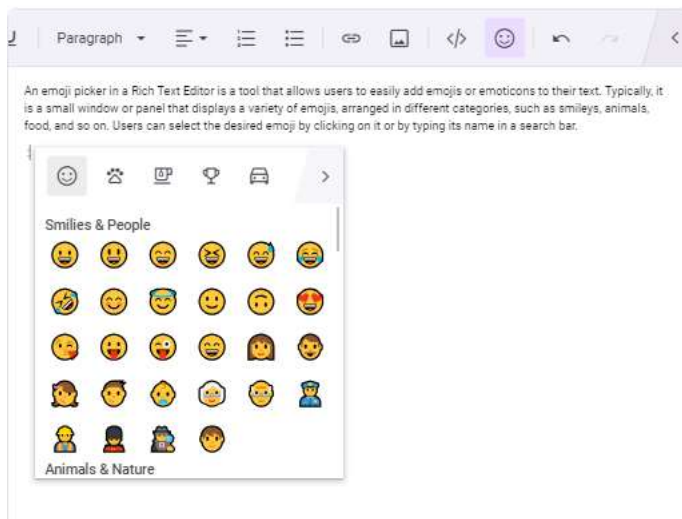
```
@Html.EJS().RichTextEditor("emojiPickerRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>
 <p>An emoji picker in a Rich Text Editor is a tool that allows users to
 easily add emojis or emoticons to their text.</p>
 <p>Typically, it is a small window or panel that displays a variety of
 emojis, arranged in different categories, such as smileys, animals, food,
 and so on. Users can select the desired emoji by clicking on it or by typing
 its name in a search bar.</p>
</div>).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "EmojiPicker", "ClearFormat", "Bold",
 "Italic", "Underline", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList", "|",
 "CreateLink", "Image", "|",
 "SourceCode", "Undo", "Redo"};
 return View();
 }
}
```

### Using the shortcut key to open the emoji picker

Quickly access the emoji picker by pressing the colon (:) key while typing a word prefix in an editor, allowing instant emoji selection and display. Moreover, continue typing in the editor after the colon (:) to filter and refine your search for the desired emojis.



### Navigating and selecting emojis using the keyboard

The emoji picker popup offers keyboard navigation options, allowing you to move the emoji focus from one emoji to another. The following keys are used for navigation:

**Arrow keys:** Use the arrow keys (up, down, left, right) to move the emoji focus in the corresponding direction.

**Enter:** Press Enter key to select the currently focused emoji.

**Escape:** Press Escape to close the emoji picker popup without selecting an emoji.

## Markdown

In Rich Text Editor, you click the toolbar buttons to format the words and the changes are visible immediately.

Markdown is not like that. When you format the word in Markdown format, you need to add Markdown syntax to the word to indicate which words and phrases should look different from each other.

Rich Text Editor supports markdown editing when the [EditorMode](#) set as **Markdown** and using both *keyboard interaction* and *toolbar action*, you can apply the formatting to text.

## Supported Commands

The ASP MVC Markdown editor supports the following commands to format the markdown content:

| Commands                | Syntax                                                                            | Description                                                                                                |
|-------------------------|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
|                         | ----- ----- -----                                                                 |                                                                                                            |
| Bold                    | Sample content for <b>bold text</b> .                                             | For bold, add <b>or</b> to front and back of the text. For order list, precede each line with a number.    |
| Italic                  | Sample content for <i>Italic text</i> .                                           | For Italic, add <i>* or _</i> to front and back of the text.                                               |
| Bold and Italics        | Sample content for <b><i>bold and Italic text</i></b> .                           | For bold and Italics, add <i>* to the front and back of the text.</i>                                      |
| Heading 1               | <b>#</b> Heading 1 content                                                        | For heading 1, add <b>#</b> to start of the line.                                                          |
| Heading 2               | <b>##</b> Heading 2 content                                                       | For heading 2, add <b>##</b> to start of the line.                                                         |
| Heading 3               | <b>###</b> Heading 3 content                                                      | For heading 3, add <b>###</b> to start of the line.                                                        |
| Heading 4               | <b>####</b> Heading 4 content                                                     | For heading 4, add <b>####</b> to start of the line.                                                       |
| Heading 5               | <b>#####</b> Heading 5 content                                                    | For heading 5, add <b>#####</b> to start of the line.                                                      |
| Heading 6               | <b>#####</b> Heading 6 content                                                    | For heading 6, add <b>#####</b> to start of the line.                                                      |
| Line Break              | First line <b>&lt;br&gt;</b> Second line                                          | For line break, press enter two times (or) add <b>&lt;br&gt;</b> in between the first and the second line. |
| Blockquotes             | <b>&gt;</b> Blockquotes text                                                      | For blockquotes, add <b>&gt;</b> to start of the line.                                                     |
| Strike Through          | Sample content for <del>strike through text</del> .                               | For strike through, add <b>~~</b> to front and back of the text.                                           |
| Code (Single line)      | <b>\Single line code\</b>                                                         | For single line code, add <b>`</b> to front and back of the text.                                          |
| Code block (Multi Line) | <b>\\&lt;br&gt;Multi line code text&lt;br&gt;Multi line code text&lt;br&gt;\\</b> | For multiple line code, add <b>\\`</b> in the new line before and after the content.                       |

| Subscript | `<sub>Subscript text</sub>` | For subscript, add `<sub>` to the front and `</sub>` to the back of the text. |

| Superscript | `<sup>Superscript text</sup>` | For superscript, add `<sup>` to the front and `</sup>` to the back of the text. |

| Ordered List | `1. First<br>1. Second` | For ordered list, preceding one or more lines of text with `1.` |

| Unordered List | `First<br>second` | For unordered list, preceding one or more lines of text with `*`. |

| Links | **Link text without title text**`<br>` `Link text ` `<br>Link text with title text<br>[ Link text ](URL , "title text")` | Create an inline link by wrapping link text in brackets `[ ]`, and then wrapping the URL as first parameter and title as second parameter in the parentheses `()`.  
**Note:** The title text is optional, if needed it can be given manually. |

| Table | | Heading 1 | Heading 2 |`<br>|-----|-----|<br>| Col A1 | Col A2 |<br>| Col B1 | Col B2 |` | Create a table using the pipes and underscores as given in the syntax to create 2 x 2 table. |

| Horizontal Line | ***(three asterix in new line)<br>(or)<br>(three underscores in new line)*** | **For horizontal line, add *or* to the start of the new line.** |

| Image | `` | Create an image by wrapping the image source in parentheses `()`. |

| Image with alternate text | `![ alternate text ](URL path)` | Create an image with alternate text by wrapping an alternative text in brackets `[]`, and then link of the image source in parentheses `()`.  
**Note:** When inserting the image using toolbar, the alternate text cannot be provided that needs to be given manually. |

| Escape tick marks supported | Sample text content with **bold and not bold text can be in the same line.** | In the syntax, the whole content is made as bold where the content not bold can be made as normal text by adding the bold syntax to the start and end of the respective text. Likewise you can do the same for various inline commands. |

| Escape Character | `\(any syntax)` | Escape any markdown syntax by prefix `\` to the syntax.  
Example:`<br>\Bold text|`

| HTML Entities | Copyright: `&#169;` - `&#169;`; `<br>`Trade mark: `&#8482;` - `&#8482;`; `<br>`Registered: `&#174;` - `&#174;`; `<br>`Ampersand: `&#38;` - `&#38;`; `<br>`Less than: `<` - `<<br>`Greater than: `>` - `>>` | For HTML entities, add `&` and `;` to the front and back of the respective entities. |

**Note:** The above listed commands alone are supported in Syncfusion Markdown editor. For other unsupported commands, you can achieve using the HTML tags in Markdown editor. The foot notes, definitions, math, and check list markdown syntax are also not supported.

### Markdown to HTML

The Rich Text Editor allows you to preview markdown changes immediately using preview. The third-party library [Marked](#) is used in this sample to convert markdown into HTML content.

This sample demonstrates how to preview markdown changes in Rich Text Editor. Type or edit the display text and apply format to view the preview of markdown.

**CSHTML**

```

@using Syncfusion.EJ2.RichTextEditor
<div class="control-wrapper">
 <div class="control-section">

@Html.EJS().RichTextEditor("markdown").EditorMode(EditorMode.Markdown).Toolb
arSettings(e =>
e.Items((object)ViewBag.items)).Value((string)ViewBag.value).Created("create
d").Render()
 </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
<script type="text/javascript">
 var mdsource, defaultRTE, textArea;
 function created() {
 defaultRTE = this;
 textArea = defaultRTE.contentModule.getEditPanel();
 textArea.addEventListener('keyup', (event) => {
markDownConversion(); });
 var rteObj = defaultRTE;
 mdsource = document.getElementById('preview-code');
 mdsource.addEventListener('click', (event) => {
 fullPreview({ mode: true, type: 'preview' });
 if ((event.currentTarget).classList.contains('e-active')) {
 defaultRTE.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
 'Formats', 'OrderedList', 'UnorderedList', '|',
 'CreateLink', 'Image', 'Undo', 'Redo']);
 } else {
 defaultRTE.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
 'Formats', 'OrderedList', 'UnorderedList', '|',
 'CreateLink', 'Image', 'Undo', 'Redo']);
 }
 });
 };
 function markDownConversion() {
 if (mdsource.classList.contains('e-active')) {
 var id = defaultRTE.getID() + 'html-preview';
 var htmlPreview = defaultRTE.element.querySelector('#' + id);
 var rteElement = defaultRTE.contentModule.getEditPanel();
 var rteValue = rteElement.value;
 htmlPreview.innerHTML = marked(defaultRTE.value);
 }
 };
 function fullPreview(event) {
 var id = defaultRTE.getID() + 'html-preview';
 htmlPreview = defaultRTE.element.querySelector('#' + id);
 if (mdsource.classList.contains('e-active')) {
 mdsource.classList.remove('e-active');
 mdsource.parentElement.title = 'Preview';
 textArea.style.display = 'block';
 textArea.style.width = '100%';
 htmlPreview.style.display = 'none';
 } else {
 mdsource.classList.add('e-active');

```

```

 if (!htmlPreview) {
 htmlPreview = ej.base.createElement('div', { className: 'e-
content' });
 htmlPreview.id = id;
 textArea.parentNode.appendChild(htmlPreview);
 }
 if (event.type === 'preview') {
 textArea.style.display = 'none';
 htmlPreview.classList.add('e-pre-source');
 } else {
 htmlPreview.classList.remove('e-pre-source');
 textArea.style.width = '50%';
 }
 htmlPreview.style.display = 'block';
 markDownConversion();
 mdsource.parentElement.title = 'Code View';
 }
}
</script>
<style>
.e-richtexteditor textarea.e-content {
 float: left;
 border-right: 1px solid rgba(0, 0, 0, 0.12);
}
.e-richtexteditor .e-rte-content .e-content {
 min-height: 150px;
}
.e-richtexteditor .e-rte-content {
 overflow: hidden;
}
.e-icon-btn.e-active .e-md-preview::before {
 content: '\e350';
}
.e-icon-btn .e-md-preview::before {
 content: '\e345';
}
.e-rte-content .e-content {
 float: right;
 overflow: auto;
 height: inherit;
 padding: 8px;
 height: 100%;
}
.e-rte-content .e-content.e-pre-source {
 width: 100%;
}
</style>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 object tools1 = new
 {

```

```

 tooltipText = "Preview",
 template = "<button id='preview-code' class='e-tbar-btn e-control e-btn e-icon-btn'>" +
 "</button>"
 };
 ViewBag.items = new[] { "Bold", "Italic", "StrikeThrough", "|",
 "Formats", "OrderedList", "UnorderedList", "|", "CreateLink", "Image",
 "CreateTable", "|", tools1, "|", "Undo", "Redo" };
 ViewBag.value = @"In Rich Text Editor , you click the toolbar
 buttons to format the words and the changes are visible immediately.
 Markdown is not like that. When you format the word in Markdown format, you
 need to add Markdown syntax to the word to indicate which words
 and phrases should look different from each other.
 RichTextEditor supports markdown editing when the editorMode set as
 markdown and using both *keyboard interaction* and *toolbar action*, you
 can apply the formatting to text.
 We can add our own custom formation syntax for the Markdown formation,
 [sample link] (https://ej2.syncfusion.com/javascript/demos/#/material/rich-
 text-editor/markdown-editor-custom-format.html) .
 The third-party library Marked is used in this sample to convert
 markdown into HTML content";
 return View();
}
}

```

## Table

Rich Text Editor allows to insert Markdown table in edit panel with 2 X 2 rows and columns along with the heading.

To use table tool, add the **CreateTable** item in toolbar items.

### Insert Table

To insert the table in Rich Text Editor, click the **table** toolbar option to insert the table into Rich Text Editor content and this is the default way in all the devices. Refer the below sample and code snippets to add the table in Markdown editor.

## CSHTML

```

@using Syncfusion.EJ2.RichTextEditor
<div class="control-wrapper">
 <div class="control-section">

@Html.EJS().RichTextEditor("markdown").EditorMode(EditorMode.Markdown).Toolb
arSettings(e =>
e.Items((object)ViewBag.items)).Value((string)ViewBag.value).Created("create
d").Render()
 </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
<script type="text/javascript">
 var mdsources, defaultRTE, textArea;
 function created() {
 defaultRTE = this;
 textArea = defaultRTE.contentModule.getEditPanel();
 textArea.addEventListener('keyup', (event) => {
 markDownConversion();
 });
 }

```



```

var rteObj = defaultRTE;
mdsource = document.getElementById('preview-code');
mdsource.addEventListener('click', (event) => {
 fullPreview({ mode: true, type: 'preview' });
 if ((event.currentTarget).classList.contains('e-active')) {
 defaultRTE.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
 'Formats', 'OrderedList', 'UnorderedList', '|',
 'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
 } else {
 defaultRTE.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
 'Formats', 'OrderedList', 'UnorderedList', '|',
 'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
 }
});
};
function markDownConversion() {
 if (mdsource.classList.contains('e-active')) {
 var id = defaultRTE.getID() + 'html-preview';
 var htmlPreview = defaultRTE.element.querySelector('#' + id);
 var rteElement = defaultRTE.contentModule.getEditPanel();
 var rteValue = rteElement.value;
 htmlPreview.innerHTML = marked(defaultRTE.value);
 }
};
function fullPreview(event) {
 var id = defaultRTE.getID() + 'html-preview';
 htmlPreview = defaultRTE.element.querySelector('#' + id);
 if (mdsource.classList.contains('e-active')) {
 mdsource.classList.remove('e-active');
 mdsource.parentElement.title = 'Preview';
 textArea.style.display = 'block';
 textArea.style.width = '100%';
 htmlPreview.style.display = 'none';
 } else {
 mdsource.classList.add('e-active');
 if (!htmlPreview) {
 htmlPreview = ej.base.createElement('div', { className: 'e-
content' });
 htmlPreview.id = id;
 textArea.parentNode.appendChild(htmlPreview);
 }
 if (event.type === 'preview') {
 textArea.style.display = 'none';
 htmlPreview.classList.add('e-pre-source');
 } else {
 htmlPreview.classList.remove('e-pre-source');
 textArea.style.width = '50%';
 }
 htmlPreview.style.display = 'block';
 markDownConversion();
 mdsource.parentElement.title = 'Code View';
 }
}
</script>
<style>

```

```

.e-richtexteditor textarea.e-content {
 float: left;
 border-right: 1px solid rgba(0, 0, 0, 0.12);
}
.e-richtexteditor .e-rte-content .e-content {
 min-height: 150px;
}
.e-richtexteditor .e-rte-content {
 overflow: hidden;
}
.e-icon-btn.e-active .e-md-preview::before {
 content: '\e350';
}
.e-icon-btn .e-md-preview::before {
 content: '\e345';
}
.e-rte-content .e-content {
 float: right;
 overflow: auto;
 height: inherit;
 padding: 8px;
 height: 100%;
}
.e-rte-content .e-content.e-pre-source {
 width: 100%;
}
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 object tools1 = new
 {
 tooltipText = "Preview",
 template = "<button id='preview-code' class='e-tbar-btn e-
control e-btn e-icon-btn'>" +
"</button>"
 };
 ViewBag.items = new[] { "Bold", "Italic", "StrikeThrough", "|",
"Formats", "OrderedList", "UnorderedList", "|", "CreateLink", "Image",
"CreateTable", "|", tools1, "|", "Undo", "Redo" };
 ViewBag.value = @"In Rich Text Editor , you click the toolbar
buttons to format the words and the changes are visible immediately.
Markdown is not like that. When you format the word in Markdown format, you
need to add Markdown syntax to the word to indicate which words
and phrases should look different from each other.
RichTextEditor supports markdown editing when the editorMode set as
markdown and using both *keyboard interaction* and *toolbar action*, you
can apply the formatting to text.
We can add our own custom formation syntax for the Markdown formation.
The third-party library Marked is used in this sample to convert
markdown into HTML content";
 return View();
 }
}

```

```
}
}
```

### Changing Table Constants

The Markdown table constants can be changed for the table heading and the column names.

### CSHTML

```
<div class="control-wrapper">
 <div class="control-section">
 @Html.EJS().RichTextEditor("markdown").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).Value((string)ViewBag.value).Created("create
d").Render()
 </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/marked/marked.min.js"></script>
<script>
 L10n.load({
 'en-US': {
 'richtexteditor': {
 'TableHeadingText': 'Header',
 'TableColText': 'Cell'
 }
 }
 });
</script>
<script type="text/javascript">
 var mdsource, defaultRTE, textArea;
 function created() {
 defaultRTE = this;
 textArea = defaultRTE.contentModule.getEditPanel();
 textArea.addEventListener('keyup', (event) => {
markDownConversion(); });
 var rteObj = defaultRTE;
 mdsource = document.getElementById('preview-code');
 mdsource.addEventListener('click', (event) => {
 fullPreview({ mode: true, type: 'preview' });
 if ((event.currentTarget.classList.contains('e-active')) {
 defaultRTE.disableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
 'Formats', 'OrderedList', 'UnorderedList', '|',
 'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
 } else {
 defaultRTE.enableToolbarItem(['Bold', 'Italic',
'StrikeThrough', '|',
 'Formats', 'OrderedList', 'UnorderedList', '|',
 'CreateLink', 'Image', 'Undo', 'Redo', 'CreateTable']);
 }
 });
 };
 function markDownConversion() {
 if (mdsource.classList.contains('e-active')) {
 var id = defaultRTE.getID() + 'html-preview';
 var htmlPreview = defaultRTE.element.querySelector('#' + id);
 var rteElement = defaultRTE.contentModule.getEditPanel();
 var rteValue = rteElement.value;
```

```

 htmlPreview.innerHTML = marked(defaultRTE.value);
 }
};
function fullPreview(event) {
 var id = defaultRTE.getID() + 'html-preview';
 htmlPreview = defaultRTE.element.querySelector('#' + id);
 if (mdsource.classList.contains('e-active')) {
 mdsource.classList.remove('e-active');
 mdsource.parentElement.title = 'Preview';
 textArea.style.display = 'block';
 textArea.style.width = '100%';
 htmlPreview.style.display = 'none';
 } else {
 mdsource.classList.add('e-active');
 if (!htmlPreview) {
 htmlPreview = ej.base.createElement('div', { className: 'e-content' });
 htmlPreview.id = id;
 textArea.parentNode.appendChild(htmlPreview);
 }
 if (event.type === 'preview') {
 textArea.style.display = 'none';
 htmlPreview.classList.add('e-pre-source');
 } else {
 htmlPreview.classList.remove('e-pre-source');
 textArea.style.width = '50%';
 }
 htmlPreview.style.display = 'block';
 markDownConversion();
 mdsource.parentElement.title = 'Code View';
 }
}
</script>
<style>
.e-richtexteditor textarea.e-content {
 float: left;
 border-right: 1px solid rgba(0, 0, 0, 0.12);
}
.e-richtexteditor .e-rte-content .e-content {
 min-height: 150px;
}
.e-richtexteditor .e-rte-content {
 overflow: hidden;
}
.e-icon-btn.e-active .e-md-preview::before {
 content: '\e350';
}
.e-icon-btn .e-md-preview::before {
 content: '\e345';
}
.e-rte-content .e-content {
 float: right;
 overflow: auto;
 height: inherit;
 padding: 8px;
 height: 100%;
}

```

```
.e-rte-content .e-content.e-pre-source {
 width: 100%;
}
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 object tools1 = new
 {
 tooltipText = "Preview",
 template = "<button id='preview-code' class='e-tbar-btn e-control e-btn e-icon-btn'>" +
 "</button>"
 };
 ViewBag.items = new[] { "Bold", "Italic", "StrikeThrough", "|",
 "Formats", "OrderedList", "UnorderedList", "|", "CreateLink", "Image",
 "CreateTable", "|", tools1, "|", "Undo", "Redo" };
 ViewBag.value = @"In Rich Text Editor , you click the toolbar
 buttons to format the words and the changes are visible immediately.
 Markdown is not like that. When you format the word in Markdown format, you
 need to add Markdown syntax to the word to indicate which words
 and phrases should look different from each other.
 Rich Text Editor supports markdown editing when the editorMode set as
 markdown and using both *keyboard interaction* and *toolbar action*, you
 can apply the formatting to text.
 We can add our own custom formation syntax for the Markdown formation.
 The third-party library Marked is used in this sample to convert
 markdown into HTML content";
 return View();
 }
}
```

See Also

- [How to integrate the third party library](#)
- [How to change the editor mode](#)

### File Browser

Rich Text Editor allows to browse and insert images in the edit panel using the file browser. File browser allows the users to browse and select a file or folder from the file system and it supports various cloud services.

The following example explains how to configure the file browser within the Rich Text Editor component.

- Configure the `FileManager` toolbar item in the `ToolbarSettings` API `Items` property.
- Set `Enable` property as `true` on `FileManagerSettings` property to make the file browser in the Rich Text Editor to appear on the `FileManager` toolbar click action.

**CSHTML**

```
@Html.EJS().RichTextEditor("fileBrowser").ToolBarSettings(e =>
e.Items((object)ViewBag.items)).FileManagerSettings(e => { e.Enable(true);
e.Path("/Pictures/Food"); e.AjaxSettings((object)ViewBag.ajaxSettings);
}).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Render()
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 string hostUrl = "https://ej2-aspcore-service.azurewebsites.net/";
 ViewBag.ajaxSettings = new {
 url = hostUrl + "api/FileManager/FileOperations",
 getImageUrl = hostUrl + "api/FileManager/GetImage",
 uploadUrl = hostUrl + "api/FileManager/Upload",
 downloadUrl = hostUrl + "api/FileManager/Download"
 };
 ViewBag.items = new[] { "FileManager", "Image" };
 return View();
 }
}
```

**Format Painter in ASP.NET MVC Rich Text Editor Control | Syncfusion**

A format painter is a tool that allows you to copy the formatting from a piece of text and apply it to another one. Format Painter can be accessed via the toolbar or the keyboard shortcuts. The format painter can copy the formatting of a single word or a whole paragraph. The format painter can be customized using the [FormatPainterSettings](#) property.

### Enabling the toolbar option for Format Painter

You can add the **FormatPainter** tool in the Rich Text Editor using the **ToolbarSettings** [Items](#) property.

By double-clicking the format painter toolbar button, **sticky mode** will be enabled. In sticky mode, the format painter will be disabled when the user clicks the **Escape** key again.

The following code example shows how to add the format painter tool in the Rich Text Editor.

#### CSHTML

```
@Html.EJS().RichTextEditor("formatPainterRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ContentTemplate(@<div>
 <h3>Format Painter</h3>
 <p>
 A Format Painter is a Rich Text Editor feature allowing users to
 quickly
 <span style="background-color: rgb(198, 140,
83);">copy
 and
 <span style="background-color: rgb(198, 140,
83);">paste
 formatting from one text to another. With a rich text editor,
 utilize the format painter as follows:
 </p>

 Select the text whose format you want to copy.

 Click on the Format Painter button in
 the toolbar. It may look like a paintbrush icon.

 The cursor will change to a paintbrush icon.
 Click and drag the cursor over the text you want to apply the copied format.

 Release the mouse button to apply the format.

 <p>
 Using the format painter in a rich text editor can save you time
 when formatting a large document, You can quickly
 copy and apply formatting
 to <span style="background-color: rgb(198, 140,
83);">multiple sections.
 It's a helpful tool for anyone who works with text editing
 regularly, such as writers, editors, and content creators.
 </p>
</div>).Render()
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
```

```

{
 ViewBag.items = new[] { "FormatPainter", "ClearFormat", "Bold",
 "Italic", "Underline", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList", "|",
 "CreateLink", "Image", "|",
 "SourceCode", "Undo", "Redo"};
 return View();
}

```

### Customization of copy and paste format

You can customize the format painter tool in the Rich Text Editor using the `FormatPainterSettings` property.

The `AllowedFormats` property helps you to specify tag names that allow the formats to be copied from the selected text. For instance, you can include formats from the selected text using tags like `p`; `h1`; `h2`; `h3`; `div`; `ul`; `ol`; `li`; `span`; `strong`; `em`; `code`; . The following example demonstrates how to customize this functionality.

Similarly, with the `DeniedFormats` property, you can utilize the selectors to prevent specific formats from being pasted onto the selected text. The table below illustrates the selectors and their respective usage.

| Type | Description        | Selector                      | Usage                                                                |
|------|--------------------|-------------------------------|----------------------------------------------------------------------|
| ( )  | Class Selector     | h3(e-rte-block-blue-text)     | The class name e-rte-block-blue-text of H3 element is not copied.    |
| []   | Attribute Selector | span\[title]                  | The title attribute of span element is not copied.                   |
| { }  | Style Selector     | span{background-color, color} | The background-color and color styles of span element is not copied. |

Using the `DeniedFormats` property following styles are denied copying from the selected text such as `h3(e-rte-block-blue-text){background-color,padding}[title]`; `li{color}`; `span(e-inline-text-highlight)[title]`; `strong{color}(e-rte-strong-bg)`.

### CSHTML

```

@Html.EJS().RichTextEditor("formatPainterRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).FormatPainterSettings(e =>
{e.AllowedFormats((string)ViewBag.allowedFormats);e.DeniedFormats((string)Vi
ewBag.deniedFormats);}).ContentTemplate(@<div>
 <h3 class="e-rte-block-blue-text" title="Format Painter" style="color:
#0079f3; background-color: #eff6ff; padding: 10px;">Format
Painter</h3>
 <p>
 A Format Painter is a Rich Text Editor feature allowing users to
quickly

```



```

<span class="e-inline-text-highlight" style="color: blue;"
title="Styled by CSS Class selector">copy
and
<span class="e-inline-text-highlight" style="color: blue;"
title="Styled by CSS Class selector">paste
 formatting from one text to another. With a rich text editor,
 utilize the format painter as follows:
</p>

 <li style="color: crimson;">
 Select the text whose format you want to copy.

 <li style="color: crimson;">
 Click on the Format Painter button in
 the toolbar. It may look like a paintbrush icon.

 <li style="color: crimson;">
 The cursor will change to a paintbrush icon.
 Click and drag the cursor over the text you want to apply the copied format.

 <li style="color: crimson;">
 Release the mouse button to apply the format.

<p>
 Using the format painter in a rich text editor can save you time
 when formatting a large document, You can quickly
 copy and apply formatting
 to <strong class="e-rte-strong-bg" style="color: blue">multiple
 sections.
 It's a helpful tool for anyone who works with text editing
 regularly, such as writers, editors, and content creators.
</p>
</div>).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "FormatPainter", "ClearFormat", "Bold",
 "Italic", "Underline", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList", "|",
 "CreateLink", "Image", "|",
 "SourceCode", "Undo", "Redo"};
 ViewBag.allowedFormats =
 "p;h1;h2;h3;div;ul;ol;li;span;strong;em;code;";
 ViewBag.deniedFormats = "h3(e-rte-block-blue-text){background-
 color,padding,color}[title]; li{color}; span(e-inline-text-
 highlight){color}[title]; strong{color}(e-rte-strong-bg);";
 return View();
 }
}

```

### Using the shortcut key to copy and paste the format

You can use the following shortcut keys to copy and paste the format in the Rich Text Editor.

| Actions          | Keyboard shortcuts | Description                                                      |
|------------------|--------------------|------------------------------------------------------------------|
| Copy the format  | Alt + Shift + c    | Copy the selection format or current range.                      |
| Paste the format | Alt + Shift + v    | Paint the copied format.                                         |
| Escape           | Esc                | Remove the previously copied format and disable the sticky mode. |

The format painter retains the formatting after application making it possible to apply the same formatting multiple times by using the Alt + Shift + v keyboard shortcut.

Additionally, You can perform the format painter actions programmatically using the [executeCommand](#) public method.

### Form support

This below sample demonstrate how to get the Rich Text Editor value in button click.

#### Render the Rich Text Editor

Render the Rich Text Editor in form as below.

```
`html
<form id="myForm" class="form-vertical">
<div class="form-group">
@Html.EJS().RichTextEditor("defaultRTE").MaxLength(100).Render()
<div id="dateError" style="padding-top: 10px"></div>
</div>
<div style="text-align: center">
<button id="validateSubmit" class="samplebtn e-control e-btn" type="submit" data-
ripple="true">Submit</button>
<button id="resetbtn" class="samplebtn e-control e-btn" type="reset" data-
ripple="true">Reset</button>
</div>
</form>
`
```

#### Obtain the Value

Upon submitting the form, `getValue` method will be triggered. Through the `FormData` class, Rich Text Editor value obtained as below.

#### CSHTML

```

<div id="content" class="box-form" style="margin: 0 auto; width:750px;
padding:25px">
 <form id="form-element" class="form-vertical">
 <div class="form-group">
 @Html.EJS().RichTextEditor("form-
support").ShowCharCount(true).MaxLength(100).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
 Users can format their content using standard toolbar commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

 </div>).Created("created").Render()
 <div id="dateError" style="padding-top: 10px"></div>
 </div>
 <div class="form-group">
 <div class="col-sm-3 control-label"></div>
 <div class="col-sm-6">
 <div id='error'></div>
 </div>
 </div>
 <div style="text-align: center">
 <button id="validateSubmit" class="samplebtn e-control e-btn"
type="submit" data-ripple="true">Submit</button>
 <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
 </div>
 </form>
</div>
<script>
 var defaultRTE, formObject, options;
 function created() {
 options = {
 rules: {
 'rteForm': {
 required: true
 }
 }
 };
 };

```

```

 formObject = new ej.inputs.FormValidator('#form-element', options);
 defaultRTE = this;
 this.element.firstChild.setAttribute("required", "");
 this.element.firstChild.setAttribute('data-required-message', '*
This field is required');
 this.element.firstChild.setAttribute('data-msg-containerid',
'dateError');
 this.element.firstChild.setAttribute('name', 'rteForm');
 };
 window.onload = function () {
 document.getElementById('validateSubmit').onclick = function () {
 getValue();
 };
 }
 function getValue() {
 formObject.validate();
 }
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### See Also

- [How to integrate the third party library](#)
- [How to validate the value](#)

### Validation

Validate the Rich Text Editor's value on form submission by applying Validation Rules and Validation Message to the Rich Text Editor.

#### Validation Rules

The Rich Text Editor is a textarea control. The Rich Text Editor also provides the functionality of character count and its validation. So, you can validate the Rich Text Editor's value on form submission by applying Validation Rules and Validation Message to the Rich Text Editor.

| Rules     | Description                                                 |
|-----------|-------------------------------------------------------------|
| required  | Requires value for the Rich Text Editor control.            |
| minlength | Requires the value to be of given minimum characters count. |
| maxlength | Requires the value to be of given maximum characters count. |

This sample is used to validate form using the obtrusive Validation. Type the values in Rich Text Editor and the form enables the validation with the formvalidator rules by clicking on the submit externally. All rules are validated by the formvalidator rules.

### CSHTML

```
<div class="control-section">
 <div id="content" class="box-form" style="margin: 0 auto; width:750px;
padding:25px">
 <form id="form-element" class="form-vertical">
 <div class="form-group">

@Html.EJS().RichTextEditor("defaultRTE").ShowCharCount(true).MaxLength(100).
Placeholder("Type something").Created("created").Change("onChange").Render()

 <div id="dateError" style="padding-top: 10px"></div>
 </div>
 <div style="text-align: center">

@Html.EJS().Button("validateSubmit").Disabled("true").Render()
 <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
 </div>
 </form>
 </div>
</div>
<script>
 var defaultRTE, formObject, options;
 function created() {
 options = {
 rules: {
 'rteForm': {
 required: true
 }
 }
 };
 formObject = new ej.inputs.FormValidator('#form-element', options);
 defaultRTE = this;
 this.element.firstChild.setAttribute("required", "");
 this.element.firstChild.setAttribute('data-msg-containerid',
'dateError');
 this.element.firstChild.setAttribute('name', 'rteForm');
 };
 window.onload = function () {
 document.getElementById('validateSubmit').onclick = function () {
 getValue();
 };
 }
 function getValue() {
 formObject.validate();
 }
 function onChange() {
 var submitButton =
document.getElementById("validateSubmit").ej2_instances[0];
 submitButton.disabled = false;
 }
</script>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

## Validation Message

The default error message for a rule can be customizable by defining it along with concern rule object as follows

**CSHTML**

```
<div class="control-section">
 <div id="content" style="margin: 0 auto; width:750px; padding:25px">
 <form id="form-element" class="form-vertical">
 <div class="form-group">
 @Html.EJS().RichTextEditor("form-
support").ShowCharCount(true).MaxLength(100).Placeholder("Type
something").Created("created").Change("onChange").Render()
 <div id="dateError" style="padding-top: 10px"></div>
 </div>
 <div style="text-align: center">

@Html.EJS().Button("validateSubmit").Disabled("true").Render()
 <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
 </div>
 </form>
 </div>
</div>
<script>
var defaultRTE, formObject, options;
function created() {
 options = {
 rules: {
 'rteForm': {
 required: true,
 minLength: [20, 'Need atleast 20 character length'],
 maxLength: [100, 'Maximum 100 character only']
 }
 }
 };
 formObject = new ej.inputs.FormValidator('#form-element', options);
 this.element.firstChild.setAttribute("required", "");
 this.element.firstChild.setAttribute('data-required-message', '*
This field is required');
 this.element.firstChild.setAttribute('data-msg-containerid',
'dateError');
 this.element.firstChild.setAttribute('name', 'rteForm');
};
document.getElementById('validateSubmit').onclick = function () {
 getValue();
}
```

```

 }
 function getValue() {
 formObject.validate();
 }
 function onChange() {
 var submitButton =
document.getElementById("validateSubmit").ej2_instances[0];
 submitButton.disabled = false;
 }
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Custom Placement of Validation Message

The FormValidator has an event `customPlacement` which can be used to place the error message from default position to desired custom location.

## CSHTML

```

<div class="control-section">
 <div id="content" class="box-form" style="margin: 0 auto; width:750px;
padding:25px">
 <form id="form-element" class="form-vertical">
 <div class="form-group">

@Html.EJS().RichTextEditor("defaultRTE").ShowCharCount(true).MaxLength(100).
Placeholder("Type something").Created("created").Change("onChange").Render()
 <div id="dateError" style="padding-top: 10px"></div>
 </div>
 <div style="text-align: center">

@Html.EJS().Button("validateSubmit").Disabled("true").Render()
 <button id="resetbtn" class="samplebtn e-control e-btn"
type="reset" data-ripple="true">Reset</button>
 </div>
 </form>
 </div>
</div>
<script>
 var defaultRTE, formObject, options;
 function created() {
 options = {
 rules: {
 'rteForm': {
 required: true,
 minLength: [6, 'Need atleast 6 character length'],
 maxLength: [100, 'Maximum 100 character only']
 }
 }
 }
 }

```

```

 },
 customPlacement: function (inputElement, error) {
 document.getElementById('dateError').appendChild(error);
 }
};
formObject = new ej.inputs.FormValidator('#form-element', options);
defaultRTE = this;
this.element.firstChild.setAttribute("required", "");
this.element.firstChild.setAttribute('data-required-message', '*
This field is required');
this.element.firstChild.setAttribute('data-msg-containerid',
'dateError');
this.element.firstChild.setAttribute('name', 'rteForm');
}
window.onload = function () {
 document.getElementById('validateSubmit').onclick = function () {
 getValue();
 };
}
function getValue() {
 formObject.validate();
}
function onChange() {
 var submitButton =
document.getElementById("validateSubmit").ej2_instances[0];
 submitButton.disabled = false;
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Globalization

### Localization

The Rich Text Editor provides an option to localize its strings; it is used to adapting the editor to a particular local language. By default, the editor will use the **US English (en-US)** as its language. Find the table with a list of keys and their corresponding values for the default language (en-US).

`typescript

```

{
 "en-US": {
 "richtexteditor": {
 "alignments": "Alignments",
 "justifyLeft": "Align Left",

```



"justifyCenter": "Align Center",  
"justifyRight": "Align Right",  
"justifyFull": "Align Justify",  
"fontName": "Font Name",  
"fontSize": "Font Size",  
"fontColor": "Font Color",  
"backgroundColor": "Background Color",  
"bold": "Bold",  
"italic": "Italic",  
"underline": "Underline",  
"strikethrough": "Strikethrough",  
"clearFormat": "Clear Format",  
"clearAll": "Clear All",  
"cut": "Cut",  
"copy": "Copy",  
"paste": "Paste",  
"unorderedList": "Bulleted List",  
"orderedList": "Numbered List",  
"indent": "Increase Indent",  
"outdent": "Decrease Indent",  
"undo": "Undo",  
"redo": "Redo",  
"superscript": "Superscript",  
"subscript": "Subscript",  
"createLink": "Insert Link",  
"openLink": "Open Link",  
"editLink": "Edit Link",  
"removeLink": "Remove Link",  
"image": "Insert Image",  
"replace": "Replace",  
"align": "Align",  
"caption": "Image Caption",  
"remove": "Remove",

```
"insertLink": "Insert Link",
"display": "Display",
"altText": "Alternative Text",
"dimension": "Change Size",
"fullscreen": "Maximize",
"maximize": "Maximize",
"minimize": "Minimize",
"lowerCase": "Lower Case",
"upperCase": "Upper Case",
"print": "Print",
"formats": "Formats",
"sourcecode": "Code View",
"preview": "Preview",
"viewside": "ViewSide",
"insertCode": "Insert Code",
"linkText": "Display Text",
"linkTooltipLabel": "Title",
"linkWebUrl": "Web Address",
"linkTitle": "Enter a title",
"linkurl": "http://example.com",
"linkOpenInNewWindow": "Open Link in New Window",
"linkHeader": "Insert Link",
"dialogInsert": "Insert",
"dialogCancel": "Cancel",
"dialogUpdate": "Update",
"imageHeader": "Insert Image",
"imageLinkHeader": "You can also provide a link from the web",
"mdimageLink": "Please provide a URL for your image",
"imageUploadMessage": "Drop image here or browse to upload",
"imageDeviceUploadMessage": "Click here to upload",
"imageAlternateText": "Alternate Text",
"alternateHeader": "Alternative Text",
"browse": "Browse",
```

"imageUrl": "http://example.com/image.png",  
"imageCaption": "Caption",  
"imageSizeHeader": "Image Size",  
"imageHeight": "Height",  
"imageWidth": "Width",  
"textPlaceholder": "Enter Text",  
"inserttablebtn": "Insert Table",  
"tabledialogHeader": "Insert Table",  
"tableWidth": "Width",  
"cellpadding": "Cell Padding",  
"cellspacing": "Cell Spacing",  
"columns": "Number of columns",  
"rows": "Number of rows",  
"tableRows": "Table Rows",  
"tableColumns": "Table Columns",  
"tableCellHorizontalAlign": "Table Cell Horizontal Align",  
"tableCellVerticalAlign": "Table Cell Vertical Align",  
"createTable": "Create Table",  
"removeTable": "Remove Table",  
"tableHeader": "Table Header",  
"tableRemove": "Table Remove",  
"tableCellBackground": "Table Cell Background",  
"tableEditProperties": "Table Edit Properties",  
"styles": "Styles",  
"insertColumnLeft": "Insert Column Left",  
"insertColumnRight": "Insert Column Right",  
"deleteColumn": "Delete Column",  
"insertRowBefore": "Insert Row Before",  
"insertRowAfter": "Insert Row After",  
"deleteRow": "Delete Row",  
"tableEditHeader": "Edit Table",  
"TableHeadingText": "Heading",  
"TableColText": "Col",

"imageInsertLinkHeader": "Insert Link",  
"editImageHeader": "Edit Image",  
"alignmentsDropDownLeft": "Align Left",  
"alignmentsDropDownCenter": "Align Center",  
"alignmentsDropDownRight": "Align Right",  
"alignmentsDropDownJustify": "Align Justify",  
"imageDisplayDropDownInline": "Inline",  
"imageDisplayDropDownBreak": "Break",  
"tableInsertRowDropDownBefore": "Insert row before",  
"tableInsertRowDropDownAfter": "Insert row after",  
"tableInsertRowDropDownDelete": "Delete row",  
"tableInsertColumnDropDownLeft": "Insert column left",  
"tableInsertColumnDropDownRight": "Insert column right",  
"tableInsertColumnDropDownDelete": "Delete column",  
"tableVerticalAlignDropDownTop": "Align Top",  
"tableVerticalAlignDropDownMiddle": "Align Middle",  
"tableVerticalAlignDropDownBottom": "Align Bottom",  
"tableStylesDropDownDashedBorder": "Dashed Borders",  
"tableStylesDropDownAlternateRows": "Alternate Rows",  
"pasteFormat": "Paste Format",  
"pasteFormatContent": "Choose the formatting action",  
"plainText": "Plain Text",  
"cleanFormat": "Clean",  
"keepFormat": "Keep",  
"formatsDropDownParagraph": "Paragraph",  
"formatsDropDownCode": "Code",  
"formatsDropDownQuotation": "Quotation",  
"formatsDropDownHeading1": "Heading 1",  
"formatsDropDownHeading2": "Heading 2",  
"formatsDropDownHeading3": "Heading 3",  
"formatsDropDownHeading4": "Heading 4",  
"fontNameSegoeUI": "SegoeUI",  
"fontNameArial": "Arial",

```

"fontNameGeorgia": "Georgia",
"fontNameImpact": "Impact",
"fontNameTahoma": "Tahoma",
"fontNameTimesNewRoman": "Times New Roman",
"fontNameVerdana": "Verdana",
"numberFormatListNumber": 'Number',
"numberFormatListLowerAlpha": 'LowerAlpha',
"numberFormatListUpperAlpha": 'UpperAlpha',
"numberFormatListLowerRoman": 'LowerRoman',
"numberFormatListUpperRoman": 'UpperRoman',
"numberFormatListLowerGreek": 'LowerGreek',
"bulletFormatListDisc": 'Disc',
"bulletFormatListCircle": 'Circle',
"bulletFormatListSquare": 'Square',
"numberFormatListNone": 'None',
"bulletFormatListNone": 'None',
"formatPainter": 'Format Painter',
"emojiPicker": 'Emoji Picker',
"embeddedCode": 'Embedded Code',
"pasteEmbeddedCodeHere": 'Paste Embedded Code here',
"emojiPickerTypeToFind": 'Type to find',
"emojiPickerNoResultFound": 'No results found',
"emojiPickerTrySomethingElse": 'Try something else',
}
}
}
,

```

To localize the editor's strings with your own localization, copy the default language informations and localize the strings in the values column. For example, to localize the editor in German language ("de-DE").

```

`typescript
{
"de-DE": {
"richtexteditor": {

```

"alignments": "Alignments",  
"justifyLeft": "Ausrichten von Text links",  
"justifyCenter": "Text-Zentrum",  
"justifyRight": "Ausrichten von Text rechts",  
"justifyFull": "rechtfertigen",  
"fontName": "Wählen Sie Schriftfamilie",  
"fontSize": "Wählen Sie Schriftgröße",  
"fontColor": "Wählen Sie die Farbe",  
"backgroundColor": "Hintergrundfarbe",  
"bold": "fett",  
"italic": "kursiv",  
"underline": "unterstreichen",  
"strikethrough": "Durchgestrichen",  
"clearAll": "Alles",  
"clearFormat": "Klar Format",  
"cut": "schneiden",  
"copy": "Kopieren",  
"paste": "Paste",  
"unorderedList": "Legen Sie ungeordnete Liste",  
"orderedList": "Geordnete Liste einfügen",  
"indent": "Einzug",  
"outdent": "Einzug verkleinern",  
"undo": "lösen",  
"redo": "Wiederherstellen",  
"superscript": "Überschrift",  
"subscript": "index",  
"createLink": "link einfügen",  
"removeLink": "fjern Hyperlink",  
"openLink": "Open link",  
"editLink": "Edit link",  
"image": "Bild einfügen",  
"replace": "ersetzen",  
"align": "ausrichten",

"caption": "Bildbeschriftung",  
"formats": "Formats",  
"remove": "Löschen",  
"insertLink": "Link einfügen",  
"display": "Anzeige",  
"alttext": "alternativer Text",  
"dimension": "Größe",  
"fullscreen": "Vollbild",  
"maximize": "Maximieren",  
"minimize": "minimieren",  
"zoomIn": "hineinzoomen",  
"zoomOut": "Rauszoomen",  
"upperCase": "Großbuchstaben",  
"lowerCase": "Kleinbuchstaben",  
"print": "Drucken",  
"sourcecode": "Quellcode",  
"preview": "Vorschau",  
"viewside": "Seite anzeigen",  
"insertcode": "Code eingeben",  
"linkText": "Displaytekst",  
"linkTooltipLabel": "tooltip",  
"linkWebUrl": "Webadres",  
"linkOpenInNewWindow": "Open de link in een nieuw venster",  
"linkHeader": "Link invoegen",  
"dialogInsert": "invoegen",  
"dialogCancel": "Annuleer",  
"dialogUpdate": "Bijwerken",  
"imageHeader": "Voeg afbeelding in",  
"imageLinkHeader": "U kunt ook een link van internet opgeven",  
"imageUploadMessage": "Zet hier een afbeelding neer of klik om te uploaden",  
"imageDeviceUploadMessage": "Klik hier om te uploaden",  
"imageAlternateText": "Alternatieve tekst",  
"alternateHeader": "Alternatieve tekst",

"browse": "Blader",  
"imageUrl": "URL",  
"imageCaption": "onderschrift",  
"imageSizeHeader": "Afbeeldingsgrootte",  
"imageHeight": "Hoogte",  
"imageWidth": "Breedte",  
"textPlaceholder": "Text eingeben",  
"inserttablebtn": "Tabelle einfügen",  
"tabledialogHeader": "Tabelle einfügen",  
"tableWidth": "Breite",  
"cellpadding": "Zellauffüllung",  
"cellspacing": "Zellabstand",  
"columns": "Anzahl der Spalten",  
"rows": "Reihenanzahl",  
"tableRows": "Tabellenzeilen",  
"tableColumns": "Tabellenspalten",  
"tableCellHorizontalAlign": "Horizontale Ausrichtung der Tabellenzelle",  
"tableCellVerticalAlign": "Vertikale Ausrichtung der Tabellenzelle",  
"createTable": "Tabelle erstellen",  
"removeTable": "Tabelle entfernen",  
"tableHeader": "Tabellenkopfzeile",  
"tableRemove": "Tabelle entfernen",  
"tableCellBackground": "Tabellenzellenhintergrund",  
"tableEditProperties": "Eigenschaften der Tabellenbearbeitung",  
"styles": "Styles",  
"insertColumnLeft": "Spalte links einfügen",  
"insertColumnRight": "Spalte rechts einfügen",  
"deleteColumn": "Spalte löschen",  
"insertRowBefore": "Zeile vor einfügen",  
"insertRowAfter": "Zeile einfügen nach",  
"deleteRow": "Zeile löschen",  
"tableEditHeader": "Tabelle bearbeiten",  
"TableHeadingText": "Überschrift",



"TableColText": "Col",  
"imageInsertLinkHeader": "Link einfügen",  
"editImageHeader": "Bild bearbeiten",  
"alignmentsDropDownLeft": "Linksbündig",  
"alignmentsDropDownCenter": "Im Zentrum anordnen",  
"alignmentsDropDownRight": "Rechts ausrichten",  
"alignmentsDropDownJustify": "Justize ausrichten",  
"imageDisplayDropDownInline": "In der Reihe",  
"imageDisplayDropDownBreak": "Brechen",  
"tableInsertRowDropDownBefore": "Reihe vorher einfügen",  
"tableInsertRowDropDownAfter": "Zeile danach einfügen",  
"tableInsertRowDropDownDelete": "Zeile löschen",  
"tableInsertColumnDropDownLeft": "Spalte links einfügen",  
"tableInsertColumnDropDownRight": "Spalte rechts einfügen",  
"tableInsertColumnDropDownDelete": "Spalte löschen",  
"tableVerticalAlignDropDownTop": "Top ausrichten",  
"tableVerticalAlignDropDownMiddle": "Mitte ausrichten",  
"tableVerticalAlignDropDownBottom": "Unten ausrichten",  
"tableStylesDropDownDashedBorder": "Gestrichelte Grenzen",  
"tableStylesDropDownAlternateRows": "Alternative Zeilen",  
"pasteFormat": "Format einfügen",  
"pasteFormatContent": "Wählen Sie die Formatierungsaktion aus",  
"plainText": "Einfacher Text",  
"cleanFormat": "sauber",  
"keepFormat": "Behalten",  
"formatsDropDownParagraph": "Absatz",  
"formatsDropDownCode": "Kodex",  
"formatsDropDownQuotation": "Zitat",  
"formatsDropDownHeading1": "Überschrift 1",  
"formatsDropDownHeading2": "Überschrift 2",  
"formatsDropDownHeading3": "Überschrift 3",  
"formatsDropDownHeading4": "Überschrift 4",  
"fontNameSegoeUI": "SegoeUI",

```

"fontNameArial": "Arial",
"fontNameGeorgia": "Georgia",
"fontNameImpact": "Einschlag",
"fontNameTahoma": "Tahoma",
"fontNameTimesNewRoman": "Mal Neu römisch",
"fontNameVerdana": "Verdana"
}
}
}
,

```

The below sample demonstrate that, the Rich Text Editor control rendered with 'de-DE' German language using [Locale](#) property.

### CSHTML

```

@Html.EJS().RichTextEditor("locale").Locale("de-DE").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Render()
<script>
 ej.base.L10n.load({
 "de-DE": {
 "richtexteditor": {
 "alignments": "Alignments",
 "justifyLeft": "Ausrichten von Text links",
 "justifyCenter": "Text-Zentrum",
 "justifyRight": "Ausrichten von Text rechts",
 "justifyFull": "rechtfertigen",

```

```
"fontName": "Wählen Sie Schriftfamilie",
"fontSize": "Wählen Sie Schriftgröße",
"fontColor": "Wählen Sie die Farbe",
"backgroundColor": "Hintergrundfarbe",
"bold": "fett",
"italic": "kursiv",
"underline": "unterstreichen",
"strikethrough": "Durchgestrichen",
"clearAll": "Alles",
"clearFormat": "Klar Format",
"cut": "schneiden",
"copy": "Kopieren",
"paste": "Paste",
"unorderedList": "Legen Sie ungeordnete Liste",
"orderedList": "Geordnete Liste einfügen",
"indent": "Einzug",
"outdent": "Einzug verkleinern",
"undo": "lösen",
"redo": "Wiederherstellen",
"superscript": "Überschrift",
"subscript": "index",
"createLink": "link einfügen",
"removeLink": "fjern Hyperlink",
"openLink": "Open link",
"editLink": "Edit link",
"image": "Bild einfügen",
"replace": "ersetzen",
"align": "ausrichten",
"caption": "Bildbeschriftung",
"formats": "Formats",
"remove": "Löschen",
"insertLink": "Link einfügen",
"display": "Anzeige",
"alttext": "alternativer Text",
"dimension": "Größe",
"fullscreen": "Vollbild",
"maximize": "Maximieren",
"minimize": "minimieren",
"zoomIn": "hineinzoomen",
"zoomOut": "Rauszoomen",
"upperCase": "Großbuchstaben",
"lowerCase": "Kleinbuchstaben",
"print": "Drucken",
"sourcecode": "Quellcode",
"preview": "Vorschau",
"viewside": "Seite anzeigen",
"insertcode": "Code eingeben",
"linkText": "Displaytekst",
"linkTooltipLabel": "tooltip",
"linkWebUrl": "Webadres",
"linkOpenInNewWindow": "Open de link in een nieuw venster",
"linkHeader": "Link invoegen",
"dialogInsert": "invoegen",
"dialogCancel": "Annuleer",
"dialogUpdate": "Bijwerken",
"imageHeader": "Voeg afbeelding in",
```

```

 "imageLinkHeader": "U kunt ook een link van internet
opgeven",
 "imageUploadMessage": "Zet hier een afbeelding neer of klik
om te uploaden",
 "imageDeviceUploadMessage": "Klik hier om te uploaden",
 "imageAlternateText": "Alternatieve tekst",
 "alternateHeader": "Alternatieve tekst",
 "browse": "Blader",
 "imageUrl": "URL",
 "imageCaption": "onderschrift",
 "imageSizeHeader": "Afbeeldingsgrootte",
 "imageHeight": "Hoogte",
 "imageWidth": "Breedte",
 "textPlaceholder": "Text eingeben",
 "inserttablebtn": "Tabelle einfügen",
 "tabledialogHeader": "Tabelle einfügen",
 "tableWidth": "Breite",
 "cellpadding": "Zellauffüllung",
 "cellspacing": "Zellabstand",
 "columns": "Anzahl der Spalten",
 "rows": "Reihenanzahl",
 "tableRows": "Tabellenzeilen",
 "tableColumns": "Tabellenspalten",
 "tableCellHorizontalAlign": "Horizontale Ausrichtung der
Tabellenzelle",
 "tableCellVerticalAlign": "Vertikale Ausrichtung der
Tabellenzelle",
 "createTable": "Tabelle erstellen",
 "removeTable": "Tabelle entfernen",
 "tableHeader": "Tabellenkopfzeile",
 "tableRemove": "Tabelle entfernen",
 "tableCellBackground": "Tabellenzellenhintergrund",
 "tableEditProperties": "Eigenschaften der
Tabellenbearbeitung",
 "styles": "Styles",
 "insertColumnLeft": "Spalte links einfügen",
 "insertColumnRight": "Spalte rechts einfügen",
 "deleteColumn": "Spalte löschen",
 "insertRowBefore": "Zeile vor einfügen",
 "insertRowAfter": "Zeile einfügen nach",
 "deleteRow": "Zeile löschen",
 "tableEditHeader": "Tabelle bearbeiten",
 "TableHeadingText": "Überschrift",
 "TableColText": "Col",
 "imageInsertLinkHeader": "Link einfügen",
 "editImageHeader": "Bild bearbeiten",
 "alignmentsDropDownLeft": "Linksbündig",
 "alignmentsDropDownCenter": "Im Zentrum anordnen",
 "alignmentsDropDownRight": "Rechts ausrichten",
 "alignmentsDropDownJustify": "Justize ausrichten",
 "imageDisplayDropDownInline": "In der Reihe",
 "imageDisplayDropDownBreak": "Brechen",
 "tableInsertRowDropDownBefore": "Reihe vorher einfügen",
 "tableInsertRowDropDownAfter": "Zeile danach einfügen",
 "tableInsertRowDropDownDelete": "Zeile löschen",
 "tableInsertColumnDropDownLeft": "Spalte links einfügen",
 "tableInsertColumnDropDownRight": "Spalte rechts einfügen",

```

```

 "tableInsertColumnDropDownDelete": "Spalte löschen",
 "tableVerticalAlignDropDownTop": "Top ausrichten",
 "tableVerticalAlignDropDownMiddle": "Mitte ausrichten",
 "tableVerticalAlignDropDownBottom": "Unten ausrichten",
 "tableStylesDropDownDashedBorder": "Gestrichelte Grenzen",
 "tableStylesDropDownAlternateRows": "Alternative Zeilen",
 "pasteFormat": "Format einfügen",
 "pasteFormatContent": "Wählen Sie die Formatierungsaktion
aus",

 "plainText": "Einfacher Text",
 "cleanFormat": "sauber",
 "keepFormat": "Behalten",
 "formatsDropDownParagraph": "Absatz",
 "formatsDropDownCode": "Kodex",
 "formatsDropDownQuotation": "Zitat",
 "formatsDropDownHeading1": "Überschrift 1",
 "formatsDropDownHeading2": "Überschrift 2",
 "formatsDropDownHeading3": "Überschrift 3",
 "formatsDropDownHeading4": "Überschrift 4",
 "fontNameSegoeUI": "SegoeUI",
 "fontNameArial": "Arial",
 "fontNameGeorgia": "Georgia",
 "fontNameImpact": "Einschlag",
 "fontNameTahoma": "Tahoma",
 "fontNameTimesNewRoman": "Mal Neu römisch",
 "fontNameVerdana": "Verdana"
 }
}
})
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## RTL

Specifies the direction of the Rich Text Editor control using the [EnableRtl](#) property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

## CSHTML

```

@Html.EJS().RichTextEditor("rtl").EnableRtl(true).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

```

```


 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

 </div>).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

**Note:** [EnableRtl](#) property will not change based on [Locale](#) property.

## XHTML validation

The editor provides an option to validate the source content of the Rich Text Editor against the XHTML standard using the 'enableXhtml' property. When you enter or modify content in the editor, it continuously checks the XHTML source content and removes elements and attributes that are not valid.

The editor checks the following settings on validation:

### Attributes

- Must be specified in lowercase.
- Proper use of quotation marks around the attributes.
- Must be valid attributes for corresponding HTML element.
- All the required attributes must be included in the HTML element.

### HTML Elements

- Must be in lowercase.
- All opening tags must be closed.
- Allows only the valid HTML elements.
- Elements must be properly nested.
- All elements must have one root element.

- Should not use inline elements inside the block elements.

### CSHTML

```
<div class="control-section">
 @Html.EJS().RichTextEditor("form-
support").EnableXhtml(true).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
</div>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Miscellaneous

#### Placeholder

Specifies the placeholder for the RichTextEditor's content used when the Rich Text Editor body is empty through the [Placeholder](#) property.

Through the `e-rte-placeholder` class we can able to define our custom font family, font color and styles to the placeholder text.

```
`css
```

```
.e-richtexteditor .e-rte-placeholder {
```

```
font-family: monospace;
}
,
```

The below sample demonstrates the **Placeholder** option in Rich Text Editor.

#### CSHTML

```
@Html.EJS().RichTextEditor("placeholder").Placeholder("Type
something").Render()
<style>
 .e-richtexteditor .rte-placeholder {
 font-family: monospace;
 }
</style>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

#### Character Count

The Rich Text Editor automatically counts the number of characters in the content area while typing using [ShowCharCount](#) property. The characters count displayed at the bottom right of the editor. Limit the number of character in the Rich Text Editor's content using [MaxLength](#) property. By default, the editor sets the characters limit value is infinity.

The character count color will be modified based on the characters in the Rich Text Editor.

| Status  | Description                                                                                                                                                                                                              |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| normal  | Till 70% of given maxLength count reach, character count color is black.                                                                                                                                                 |
| warning | Once the number of character count in the Rich Text Editor reached 70% of given maxLength count, the character count color will be orange, indicating that, the Rich Text Editor value going to reach the maximum count. |
| error   | Once the number of character count in the Rich Text Editor reached 90% of given maxLength count, the character count color will be red, indicating that, the Rich Text Editor value reached the maximum count.           |

#### CSHTML

```
@Html.EJS().RichTextEditor("characterCount").ShowCharCount(true).MaxLength(2000).ContentTemplate(@<div>
<p>
```



The Rich Text Editor control is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content.

Users can format their content using standard toolbar commands.

```

</p>
<p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
</p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Print

The editor provides print tools which use to print the contents of the editor.

## CSHTML

```

@Html.EJS().RichTextEditor("print").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>

</div>

```

```

<p> Provides HTML view to edit the source directly for
developers.</p>
<p> Supports third - party library integration.</p>
<p> Allows preview of modified content before saving
it.</p>
<p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Print" };
 return View();
 }
}

```

## Code View

Rich Text Editor includes the ability for users to directly edit HTML code via **Source View** in the text area. If you made any modification in Source view directly, the changes will be reflected in the Rich Text Editor's content. So, the users will have more flexibility over the content they have created.

This sample used [Code mirror](#) plugin helps to highlight the HTML content and when changes happens in code view, the same has been reflected in preview mode.

## CSHTML

```

<div class="control-section">
 @Html.EJS().RichTextEditor("defaultRTE").ToolBarSettings(e =>
e.Items((object)ViewBag.tools)).ContentTemplate(@<div>
 <p>The Rich Text Editor is WYSIWYG ('what you see is what you get')
editor useful to create and edit content, and return the valid HTML
markup or markd
own of the content</p>
 <p> Toolbar </p>

 <p> Toolbar contains commands to align the text, insert link,
insert image, insert list, undo / redo operations, HTML view, etc </p>

 <p> Toolbar is fully customizable </p>

 <p> Links </p>


```

```

 <p> You can insert a hyperlink with its corresponding dialog</p>

 <p> Attach a hyperlink to the displayed text. </p>

 <p> Customize the quick toolbar based on the hyperlink </p>

<p> Image.</p>

 <p> Allows you to insert images from an online source as well as
the local computer</p>

 <p> You can upload an image</p>

 <p> Provides an option to customize quick toolbar for an image
</p>

<img alt='Logo' src='https://ej2.syncfusion.com/demos/src/rich-text-
editor/images/RTEImage-Feather.png' style='width: 300px' />
</div>).ShowCharCount(true).MaxLength(2000).Created("created").ActionComple
e("actionCompleteHandler").Render()
</div>
<link
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.39.0/codemirror.cs
s" rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js"
type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js
" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js
" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/
htmlmixed.js" type="text/javascript"></script>
<style>
 .e-code-mirror::before {
 content: '\e345';
 }
 .e-html-preview::before {
 content: '\e350';
 }
 .CodeMirror-linenumber,
 .CodeMirror-gutters {
 display: none;
 }
 .sb-header {
 z-index: 100;
 }
}

```

```

</style>
<script type="text/javascript">
 var defaultRTE;
 var divPreview;
 var myCodeMirror;
 var textArea
 divPreview = document.getElementById('DIV_Preview');
 function created() {
 defaultRTE = this;
 textArea = defaultRTE.contentModule.getEditPanel();
 }
 function mirrorConversion(e) {
 var id = defaultRTE.getID() + 'mirror-view';
 var mirrorView = defaultRTE.element.querySelector('#' + id);
 var charCount = defaultRTE.element.querySelector('.e-rte-character-
count');
 if (e.targetItem === 'Preview') {
 textArea.style.display = 'block';
 mirrorView.style.display = 'none';
 textArea.innerHTML = myCodeMirror.getValue();
 charCount.style.display = 'block';
 }
 else {
 if (!mirrorView) {
 mirrorView = ej.base.createElement('div', { className: 'e-
content' });
 mirrorView.id = id;
 textArea.parentNode.appendChild(mirrorView);
 }
 else {
 mirrorView.innerHTML = '';
 }
 textArea.style.display = 'none';
 mirrorView.style.display = 'block';
 renderCodeMirror(mirrorView, defaultRTE.value);
 charCount.style.display = 'none';
 }
 }
 function renderCodeMirror(mirrorView, content) {
 myCodeMirror = CodeMirror(mirrorView, {
 value: content,
 lineNumbers: true,
 mode: 'text/html',
 lineWrapping: true,
 });
 }
 function actionCompleteHandler(e) {
 if (e.targetItem && (e.targetItem === 'SourceCode' || e.targetItem
=== 'Preview')) {
 this.sourceCodeModule.getPanel().style.display = 'none';
 mirrorConversion(e);
 }
 else {
 setTimeout(function () {
 defaultRTE.toolbarModule.refreshToolbarOverflow(); }, 400);
 }
 }
}

```

```
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "SourceCode" };
 return View();
 }
}
```

### Full Screen

Stretches the editor to the maximum width and height of the browser window through the **FullScreen** tool in the toolbar.

### CSHTML

```
@Html.EJS().RichTextEditor("fullscreen").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "FullScreen" };
 return View();
 }
}
```

```
}

```

### Prevention of cross-site scripting (XSS)

The Rich Text Editor allows the users to edit the content with security by preventing cross-site scripting (XSS). By default, provided built-in support to remove the elements from editor content, which cause XSS attack. The editor removes the elements based on the attributes if it is possible to execute script.

In the following sample, removed `script` tag and `onmouseover` attribute from content of the Rich Text Editor.

#### CSHTML

```
@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<div><div
onmouseover="javascript:alert(1)">Prevention of Cross Sit Scripting
(XSS)</div><script>alert("hi")</script></div>).Render()
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

**Note:** It is only applicable to editorMode as HTML.

#### Custom cross-site scripting

You can also filter the elements and attributes additionally, which cause the XSS attack through [BeforeSanitizeHtml](#) event. Return the value from the event argument `helper` function to apply in the editor. To prevent the built-in support and make own cross-site scripting rules, set `cancel` argument to true.

The following sample demonstrates how to filter `script` tag from value.

#### CSHTML

```
@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<div><div>Preventi
on of Cross Sit Scripting
(XSS)</div><script>alert("hi")</script></div>).BeforeSanitizeHtml("beforeSan
itizeHtml").Render()
<script>
function beforeSanitizeHtml(args) {
 args.helper = (value) => {
 args.cancel = true;
 var temp = document.createElement('div');
 temp.innerHTML = value;
 var scriptTag = temp.querySelector('script');
 if (scriptTag) {
 ej.base.detach(scriptTag);
 }
 return temp.innerHTML;
 }
}
```

```
}
</script>
```

### **CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Resizable support

This feature allows the editor to be resized dynamically. The users can enable or disable this feature using the `EnableResize` property in the Rich Text Editor. If `EnableResize` is set to true, the Rich Text Editor component creates grip at the bottom right corner, which allows resizing the component in the diagonal direction. The following code demonstrates the resizable feature.

### *Enabling the resizable support*

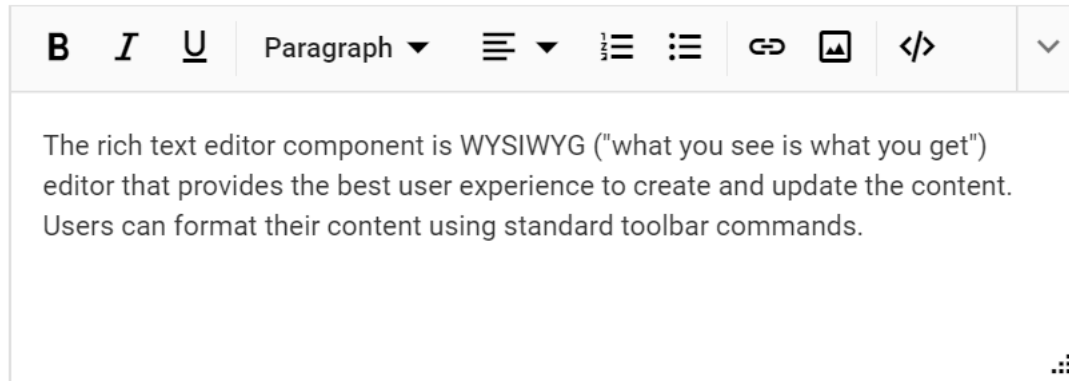
To render the Rich Text Editor in the resizable mode, set the `EnableResize` property to true.

### **CSHTML**

```
@Html.EJS().RichTextEditor("resizable").EnableResize(true).ContentTemplate(@
<div>
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
 Users can format their content using standard toolbar
commands.</p>
</div>).Render()
```

### **CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```



### *Specifying the Minimum and Maximum width and height for Resize*

To have a restricted resizable area for the Rich Text Editor, you need to specify the min-width, max-width, min-height, and max-height CSS properties for the control's wrapper element. By default, the control is capable of resizing upto the current viewport. The `e-richtexteditor` CSS class will be available in the component's wrapper and can be used for applying the above mentioned styles.

### CSHTML

```
@Html.EJS().RichTextEditor("resizable").EnableResize(true).ContentTemplate(@
<div>
<p>The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
 Users can format their content using standard toolbar
commands.</p>
 <p> Key features:</p>
</div>).Render()
<script>
 .e-richtexteditor {
 min-width: 200px;
 max-width: 800px;
 min-height: 100px;
 max-height: 300px;
 }
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Number and Bullet Format Lists

This feature allows the user to change the appearance of the Numbered and Bulleted lists. Users can also apply different numbering or bullet formats lists such as lowercase greek, upper Alpha, square and



circles. You can also customize the style type of the lists to be populated in the dropdown from the toolbar by using the `NumberFormatList` and `BulletFormatList` properties in the Rich Text Editor.

### CSHTML

```
@Html.EJS().RichTextEditor("format-lists").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).ToolbarSettings(e => e.Items((object)ViewBag.items)).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "NumberFormatList", "BulletFormatList" };
 return View();
 }
}
```

### Third party Integration

The Rich Text Editor can be integrated with third-party to suite the application scenario.

#### Code-Mirror Integration

Rich Text Editor comes with a basic HTML source editor through view-source property. [Code mirror](#) plugin can be used to highlight the syntax of HTML. CodeMirror plugin for Rich Text Editor makes editing of HTML source code with a pleasant experience.

Import necessary CSS and JS files of CodeMirror to the HTML page.

Required JS files of code mirror.

`html

```
<script src="scripts/CodeMirror/codemirror.js" type="text/javascript"></script>
<script src="scripts/CodeMirror/javascript.js" type="text/javascript"></script>
<script src="scripts/CodeMirror/css.js" type="text/javascript"></script>
<script src="scripts/CodeMirror/htmlmixed.js" type="text/javascript"></script>
```

Required CSS file of code mirror.

```
`html
```

```
<link href="scripts/CodeMirror/codemirror.min.css" rel="stylesheet" />
```

Add a custom icon for HTML source editor in the toolbar of Rich Text Editor using template option of [ToolbarSettings](#) and define the code mirror plugins, and then pass the Rich Text Editor content as argument in [ActionComplete](#) event.

### CSHTML

```
<div class="control-section">
 @Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object) ViewBag.tools)).ContentTemplate(@<div>
 <p>The Rich Text Editor is WYSIWYG ('what you see is what you get')
editor useful to create and edit content, and return the valid HTML
markup or markd
own of the content</p>
 <p> Toolbar </p>

 <p> Toolbar contains commands to align the text, insert link,
insert image, insert list, undo / redo operations, HTML view, etc </p>

 <p> Toolbar is fully customizable </p>

 <p> Links </p>

 <p> You can insert a hyperlink with its corresponding dialog</p>

 <p> Attach a hyperlink to the displayed text. </p>

 <p> Customize the quick toolbar based on the hyperlink </p>

 <p> Image.</p>


```

```

 <p> Allows you to insert images from an online source as well as
the local computer</p>

 <p> You can upload an image</p>

 <p> Provides an option to customize quick toolbar for an image
</p>

<img alt='Logo' src='https://ej2.syncfusion.com/demos/src/rich-text-
editor/images/RTEImage-Feather.png' style='width: 300px' />
</div>).ShowCharCount(true).MaxLength(2000).Created("created").ActionComple
e("actionCompleteHandler").Render()
</div>
<link
href="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.39.0/codemirror.cs
s" rel="stylesheet" />
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/codemirror.js"
type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/css/css.js
" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/xml/xml.js
" type="text/javascript"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/codemirror/5.3.0/mode/htmlmixed/
htmlmixed.js" type="text/javascript"></script>
<style>
 .e-code-mirror::before {
 content: '\e345';
 }
 .e-html-preview::before {
 content: '\e350';
 }
 .CodeMirror-linenumber,
 .CodeMirror-gutters {
 display: none;
 }
 .sb-header {
 z-index: 100;
 }
</style>
<script type="text/javascript">
 var defaultRTE;
 var divPreview;
 var myCodeMirror;
 var textArea
 divPreview = document.getElementById('DIV_Preview');
 function created() {
 defaultRTE = this;
 textArea = defaultRTE.contentModule.getEditPanel();
 }
 function mirrorConversion(e) {

```

```

var id = defaultRTE.getID() + 'mirror-view';
var mirrorView = defaultRTE.element.querySelector('#' + id);
var charCount = defaultRTE.element.querySelector('.e-rte-character-
count');
if (e.targetItem === 'Preview') {
 textArea.style.display = 'block';
 mirrorView.style.display = 'none';
 textArea.innerHTML = myCodeMirror.getValue();
 charCount.style.display = 'block';
}
else {
 if (!mirrorView) {
 mirrorView = ej.base.createElement('div', { className: 'e-
content' });
 mirrorView.id = id;
 textArea.parentNode.appendChild(mirrorView);
 }
 else {
 mirrorView.innerHTML = '';
 }
 textArea.style.display = 'none';
 mirrorView.style.display = 'block';
 renderCodeMirror(mirrorView, defaultRTE.value);
 charCount.style.display = 'none';
}
}
function renderCodeMirror(mirrorView, content) {
 myCodeMirror = CodeMirror(mirrorView, {
 value: content,
 lineNumbers: true,
 mode: 'text/html',
 lineWrapping: true,
 });
}
function actionCompleteHandler(e) {
 if (e.targetItem && (e.targetItem === 'SourceCode' || e.targetItem
=== 'Preview')) {
 this.sourceCodeModule.getPanel().style.display = 'none';
 mirrorConversion(e);
 }
 else {
 setTimeout(function () {
defaultRTE.toolbarModule.refreshToolbarOverflow(); }, 400);
 }
}
}
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "SourceCode" };
 return View();
 }
}

```

```
}

```

### Embed.ly Integration

Rich Text Editor easily integrate with [Embed.ly](#) which is probably the best service when it comes to embed the rich content such as Twitter, Facebook, Instagram and lots of other publishing platform embeds.

```
`html

```

```
<script src="https://cdn.embedly.com/widgets/platform.js" charset="UTF-8"></script>

```

In the following sample, the `Embed.ly` class `embedly-card` has been added to `<a>` tag in [ActionComplete](#) event.

### CSHTML

```
<div class="control-section">
 @Html.EJS().RichTextEditor("thirdparty").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

 </div>).ActionComplete("actionComplete").ToolbarSettings(e =>
 e.Items((object)ViewBag.items)).Render()
</div>
<script async src="https://cdn.embedly.com/widgets/platform.js"
charset="UTF-8"></script>
<script>
 function actionComplete(args) {
 if (args.requestType === 'Links') {
 if (args.elements[0].parentNode &&
 args.elements[0].parentNode.tagName === 'A') {
 var emberEle = document.createElement('blockquote');
 emberEle.setAttribute('class', 'embedly-card');
 emberEle.appendChild(args.elements[0].parentElement);
 }
 }
 }

```

```

 emberEle.appendChild(document.createElement('p'));
 args.range.insertNode(emberEle);
 }
}
}
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new object[] { "CreateLink" };
 return View();
 }
}

```

## Inline Mode

This is the inline example for the Rich Text Editor. For this you must set the [InlineMode](#) property.

Inline edition allows you to select any editable element or click the element on the page and edit it in-place.

Inline editing is a true WYSIWYG formation and on the contrary to Rich Text Editor HTML/MD editing, the styles that are used for edited content comes directly from the document stylesheet. This means that inline editors ignore the default Rich Text Editor content styles.

## Show on Select/Click

Enabling the `onSelection` option of `inlineMode` makes the inline Rich Text Editor to appear. You can select the text in the editable area otherwise the inline Rich Text Editor will be appear once click into the editable area.

## CSHTML

```

@Html.EJS().RichTextEditor("inlineMode").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>

</div>

```

```

<p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).InlineMode(e => e.Enable(true).OnSelection(true)).ToolbarSettings(e
=> e.Items((object)ViewBag.items)).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Bold", "Italic", "Underline",
"StrikeThrough", "-",
"Formats", "Alignments", "OrderedList", "UnorderedList" };
 return View();
 }
}

```

The RichTextEditor component is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content. Users can format their content using standard toolbar commands.

### Key features:



Provides the necessary functionality on demand.

- Provides a fully customizable toolbar.
- Provides HTML view to edit the source directly for developers.
- Supports third - party library integration.
- Allows preview of modified content before saving it.
- Handles images, hyperlinks, video, hyperlinks, uploads, etc.
- Contains undo / redo manager.
- Creates bulleted and numbered lists.

See Also

- [How to edit the quick toolbar settings](#)
- [How to insert link editing option in the toolbar items](#)
- [How to insert image editing option in the toolbar items](#)

## Paste from MS Word

The Rich Text Editor allows you to reduce the effort while converting the Microsoft Word content to HTML format with format and styles.

### MS Word to HTML

By default, Rich Text Editor consider the following processes on paste content from Microsoft Word.

**List conversion:** The list elements copied from the Microsoft Word document contains paragraph tags with styles and classes. The list elements are converted to standard HTML list elements by referring the styles and class names in the paragraph tags.

**Converting style:** The styles of the elements copied from the Microsoft Word document are converted to standard CSS styles and added as inline styles for each respective element.

**Tags and comments:** The Microsoft Word specific XML tags and comments are removed when cleanup on paste.

### Paste cleanup

You can control the formatting and styles on pasting the content to the editor using the pasteCleanup settings property. The following settings are available to clean up the content:

| API                               | Description                                                                                   | Default Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Type     |
|-----------------------------------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| <a href="#">prompt</a>            | To invoke prompt dialog with paste options on pasting the content in editor.                  | false                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | boolean  |
| <a href="#">plainText</a>         | To paste the content as plain text.                                                           | false                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | boolean  |
| <a href="#">keepFormat</a>        | To keep the same format with copied content.                                                  | true                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | boolean  |
| <a href="#">deniedTags</a>        | To ignore the tags when pasting HTML content.                                                 | null                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | string[] |
| <a href="#">deniedAttrs</a>       | To paste the content by filtering out these attributes from the content.                      | null                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | string[] |
| <a href="#">allowedStyleProps</a> | To paste the content by accepting these style attributes and removing other style attributes. | ['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width'] | string[] |

### Prompt dialog

When **prompt** is set to true, pasting the content in the editor will open a dialog box that contains three options **Keep**, **Clean**, and **Plain Text** as radio buttons:

1. **Keep:** Radio button to keep the same format with copied content.
2. **Clean:** Radio button to clear all the style formats with copied content.
3. **Plain Text:** Radio button to paste the copied content as plain text without any formatting or style (including the removal of all tags).



**Note:** When `prompt` value is set true, the API properties `plainText` and `keepFormat` will not be considered for processing when pasting the content.

#### Paste as plain text

When `plainText` is set to true, the copied content will be converted as plain text by removing all the HTML tags and styles applied to it and only the plain text is pasted in the editor.

**Note:** When `plainText` value is set true, the API property `prompt` should be set to false, and `keepFormat` will not be considered for processing when pasting the content.

#### Keep format

When `keepFormat` is set to true, the copied content will maintain all the style formatting allowed in the `allowedStyleProps` on pasting the content in the editor.

When `keepFormat` is set to false, the style in the copied content will be removed without considering the allowed styles in the `allowedStyleProps` when pasting the content in the editor.

**Note:** When `keepFormat` value is set true, the API property `prompt` and `plainText` should be set to false.

#### Denied tags

When `deniedTags` values are set, the tags that matches the 'denied tags' list will be removed on pasting the copied content in the editor. For Example,

1. `'a'`: Paste the content by filtering out anchor tags.
2. `'a[!href]'`: Paste the content by filtering out anchor tags that do not have the 'href' attribute.
3. `'a[href, target]'`: Paste the content by filtering out anchor tags that have the 'href' and 'target' attributes.

#### Denied attributes

When the `deniedAttrs` values are set, the attributes that matches the 'denied attributes' list will be removed on pasting the copied content in the editor. For Example,

`'id', 'title'`: This will remove the attributes 'id' and 'title' from all tags.

#### Allowed style properties

By default, the following basic styles are allowed on pasting the content to the editor.

`['background', 'background-color', 'border', 'border-bottom', 'border-left', 'border-radius', 'border-right', 'border-style', 'border-top', 'border-width', 'clear', 'color', 'cursor', 'direction', 'display', 'float', 'font', 'font-family', 'font-size', 'font-weight', 'font-style', 'height', 'left', 'line-height', 'margin', 'margin-top', 'margin-left', 'margin-right', 'margin-bottom', 'max-height', 'max-width', 'min-height', 'min-width', 'overflow', 'overflow-x', 'overflow-y', 'padding', 'padding-bottom', 'padding-left', 'padding-right', 'padding-top', 'position', 'right', 'table-layout', 'text-align', 'text-decoration', 'text-indent', 'top', 'vertical-align', 'visibility', 'white-space', 'width']`

When you configure `allowedStyleProps`, the styles, which matches the 'allowed style properties' list are allowed, all other style properties will be removed on pasting the content in the editor.

For Example,

`allowedStyleProps: ['color', 'margin']`: This will allow only the style properties 'color' and 'margin' in each pasted element.

In the following example, the paste cleanup related settings are explained with its module configuration

### CSHTML

```
@Html.EJS().RichTextEditor("pasteCleanup").ShowCharCount(true).MaxLength(1000).ContentTemplate(@<div>
 <p>RichTextEditor is a WYSIWYG editing control which will reduce the effort for users while trying to express their formatting word content as HTML format.</p>
 <p>Paste Cleanup properties:</p>

 <p>prompt - specifies whether to enable the prompt when pasting in Rich Text Editor.</p>

 <p>plainText - specifies whether to paste as plain text or not in Rich Text Editor.</p>

 <p>keepFormat- specifies whether to keep or remove the format when pasting in Rich Text Editor.</p>

 <p>deniedTags - specifies the tags to restrict when pasting in Rich Text Editor.</p>

 <p>deniedAttributes - specifies the attributes to restrict when pasting in Rich Text Editor.</p>

 <p>allowedStyleProperties - specifies the allowed style properties when pasting in Rich Text Editor.</p>

</div>).PasteCleanupSettings(p => p.Prompt(true)).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Mention Integration with Rich Text Editor

By integrating the [Mention](#) control with a Rich Text Editor, users can easily mention or tag other users or objects from the suggested list without having to manually type out their names or other identifying information.

The [Target](#) property of the Mention control allows you to specify the ID of the content editable div element within the Rich Text Editor that you want to bind the Mention control to. This allows you to enable the Mention functionality within the Rich Text Editor, so that users can mention or tag other users or objects from the suggested list while editing the text.

When the user types the @ symbol followed by a character, the Rich Text Editor will display a list of suggestions for items that the user can select from. The user can then select an item from the list by clicking on it, or by typing the name of the item they want to tag.

In the following sample, configured the following properties with popup dimensions.

- [AllowSpaces](#) - Allow to continue search action if user enter space after mention character while searching.
- [SuggestionCount](#) - The maximum number of items that will be displayed in the suggestion list.
- [ItemTemplate](#) - Used to display the customized appearance in suggestion list.

### CSHTML

```
@{
 char nameMentionChar = '@';
 List<EmployeeData> data = new EmployeeData().EmployeeList();
}
@Html.EJS().RichTextEditor("mentionIntegration").ContentTemplate(@<div>
 <p>Hello @Maria​</p>
 <p>Welcome to the mention integration with rich text editor demo. Type
 <code>@@</code> character and tag user from the suggestion list. </p>
</div>).Render()
@Html.EJS().Mention("mentionRTE").MentionChar(nameMentionChar).Target("#ment
ionIntegration_rte-edit-
view").DataSource((IEnumerable<EmployeeData>)data).AllowSpaces(true).Fields(
new Syncfusion.EJ2.DropDowns.MentionFieldSettings { Text = "Name",
Value="EmailId" }).SuggestionCount(8).Highlight(true).DisplayTemplate("@${ Name}").ItemTemplate("<table><tr><td><div
id=\"mention-TemplateList\"><img class=\"mentionEmpImage\"
src=\"./images/${EmployeeImage}\" alt=\"employee\" /><span class=\"e-badge
e-badge-success e-badge-overlap e-badge-dot e-badge-bottom
${Status}\"></div></td><td>${Name}${EmailId}</td></tr></table>").PopupWidth("250px").Pop
upHeight("200px").Render()
<style>
 #mention-TemplateList {
 position: relative;
 display: inline-block;
 padding: 2px;
 }
 .person, .email {
 display: block;
 line-height: 20px;
 text-indent: 5px;
 }
 .person {
 font-size: 16px;
 }
 .mentionEmpImage {
```

```

 display: inline-block;
 width: 46px;
 height: 46px;
 padding: 3px;
 border-radius: 25px;
 }
 #mention-TemplateList .e-badge-success {
 left: 76%;
 bottom: 4px;
 top: auto;
 }
 #mention_integration_rte-edit-view_popup .e-dropdownbase .e-list-item {
 line-height: 8px;
 }
 #mention-TemplateList .e-badge-success {
 background-color: #4d841d;
 color: #fff;
 }
 #mention-TemplateList .e-badge-success.away {
 background-color: #fedd2d;
 color: #fff;
 }
 #mention-TemplateList .e-badge-success.busy {
 background-color: #delala;
 color: #fff;
 }
 #mention-TemplateList .e-badge.e-badge-dot {
 height: 10px;
 width: 10px;
 }
 #mentionIntegration .e-mention-chip{
 cursor: pointer;
 }
</style>

```

### CONTROLLER.CS

```

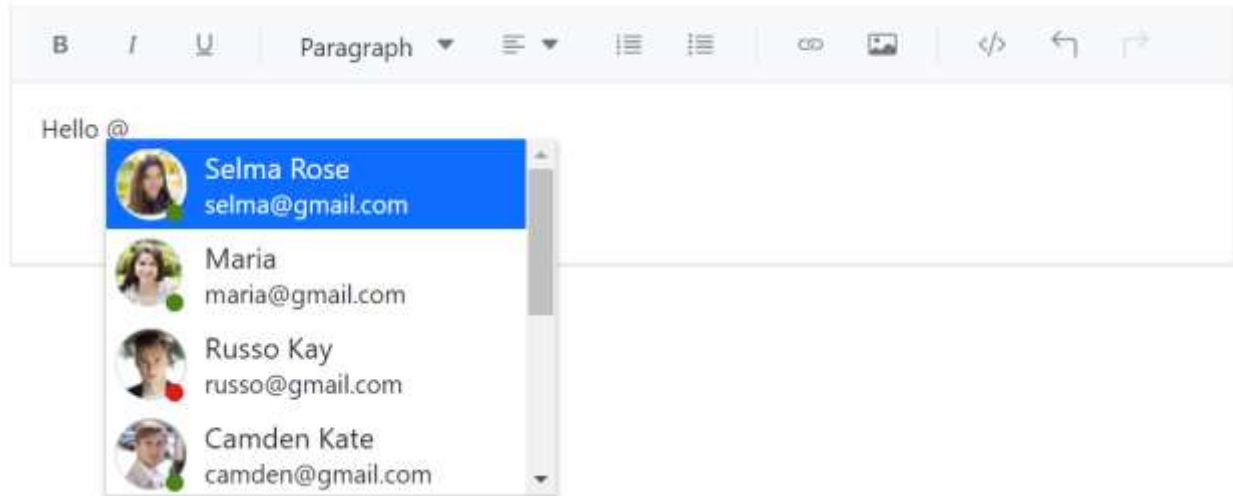
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
namespace WebApplication1.Models
{
 public class EmployeeData
 {
 public string Name { get; set; }
 public string EmployeeImage { get; set; }
 public string EmailId { get; set; }
 public List<EmployeeData> EmployeeList()
 {
 List<EmployeeData> mention = new List<EmployeeData>()
 {
 new EmployeeData { Name = "Selma Rose", EmployeeImage =
"3.png", EmailId = "selma@gmail.com" },
 new EmployeeData { Name = "Russo Kay", EmployeeImage =
"8.png", EmailId = "russo@gmail.com" },
 }
 }
 }
}

```

```

 new EmployeeData { Name = "Camden Kate", EmployeeImage =
"9.png", EmailId = "camden@gmail.com" },
 new EmployeeData { Name = "Mary Kate", EmployeeImage =
"4.png", EmailId = "mary@gmail.com" },
 new EmployeeData { Name = "Ursula Ann", EmployeeImage =
"2.png", EmailId = "ursula@gmail.com" },
 new EmployeeData { Name = "Margaret", EmployeeImage =
"5.png", EmailId = "margaret@gmail.com" },
 new EmployeeData { Name = "Laura Grace", EmployeeImage =
"6.png", EmailId = "laura@gmail.com" },
 new EmployeeData { Name = "Robert", EmployeeImage = "8.png",
EmailId = "robert@gmail.com" },
 new EmployeeData { Name = "Albert", EmployeeImage = "9.png",
EmailId = "albert@gmail.com" },
 new EmployeeData { Name = "Michale", EmployeeImage =
"5.png", EmailId = "michale@gmail.com" },
 new EmployeeData { Name = "Andrew James", EmployeeImage =
"7.png", EmailId = "james@gmail.com" },
 new EmployeeData { Name = "Rosalie", EmployeeImage =
"4.png", EmailId = "rosalie@gmail.com" },
 new EmployeeData { Name = "Stella Ruth", EmployeeImage =
"2.png", EmailId = "stella@gmail.com" },
 new EmployeeData { Name = "Richard Rose", EmployeeImage =
"8.png", EmailId = "richard@gmail.com" },
 new EmployeeData { Name = "Gabrielle", EmployeeImage =
"3.png", EmailId = "gabrielle@gmail.com" },
 new EmployeeData { Name = "Thomas", EmployeeImage = "7.png",
EmailId = "thomas@gmail.com" },
 new EmployeeData { Name = "Charles Danny", EmployeeImage =
"8.png", EmailId = "charles@gmail.com" },
 new EmployeeData { Name = "Daniel", EmployeeImage = "5.png",
EmailId = "daniel@gmail.com" },
 new EmployeeData { Name = "Matthew", EmployeeImage =
"7.png", EmailId = "matthew@gmail.com" },
 new EmployeeData { Name = "Donald Krish", EmployeeImage =
"9.png", EmailId = "donald@gmail.com" },
 new EmployeeData { Name = "Yohana", EmployeeImage = "1.png",
EmailId = "yohana@gmail.com" },
 new EmployeeData { Name = "Kevin Paul", EmployeeImage =
"5.png", EmailId = "kevin@gmail.com" },
 };
 return mention;
}
}
}

```



[View Sample](#)

See Also

- [Mention](#)

### Enter and Shift-Enter Key's Customization

Rich Text Editor allows to customize the tag that is inserted when pressing the enter key and shift + enter key in the Rich Text Editor.

#### Enter key customization

By default, the `<p>` tag will be created while pressing the enter key. The enter key can be customized by using the [EnterKey](#) property, where the possible tags that can be used to customize are `<p>`, `<div>`, and `<br>`.

When the enter key is customized with any of the possible values, pressing the enter key in the editor will create a new tag that is configured. Also, when the enter key is configured the default value of the Rich Text Editor will also change respectively with the configured values.

#### CSHTML

```
@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<p>
 In Rich text Editor, the enter key and shift + enter key actions can be
 customized using the enterKey and shiftEnterKey APIs.
</p>).EnterKey(Syncfusion.EJ2.RichTextEditor.EnterKey.DIV).Render()
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Shift-Enter key customization

By default, the `<br>` tag will be created while pressing the shift + enter key. The shift + enter key can be customized by using the [ShiftEnterKey](#) property where the possible tags that can be used to customize are `<br>`, `<p>`, `<div>`.

When the shift + enter key is customized with any of the possible values, pressing the shift + enter key in the editor will create a new tag that is configured. Also, when the shift + enter key is configured the default value of the Rich Text Editor will change respectively with the configured values.

### CSHTML

```
@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<p>
 In Rich text Editor, the enter key and shift + enter key actions can be
 customized using the enterKey and shiftEnterKey APIs.
</p>).ShiftEnterKey(Syncfusion.EJ2.RichTextEditor.ShiftEnterKey.BR).Render()
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Iframe

When the [IframeSettings](#) option is enabled, the Rich Text Editor creates the iframe element as the content area on control initialization; it is used to display and editing the content. In content area, the editor displays only the body tag of a `<iframe>` document.

### CSHTML

```
@Html.EJS().RichTextEditor("iframe").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>

</div>
```

```

 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).IframeSettings(iframeSettings =>
iframeSettings.Enable(true)).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Iframe Attributes

The editor allows you to pass an additional attribute to body tag of a <iframe> element using attributes fields of the `attributes` fields of `IframeSettings` property. This property contains name/value pairs in string format. It is used to override the default appearance of the content area.

## CSHTML

```

@Html.EJS().RichTextEditor("iframe").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Created("created").IframeSettings(iframeSettings =>
iframeSettings.Enable(true)).Render()
<script>
 function created() {
 this.setProperties({
 iframeSettings: {
 attributes: {

```



```

 readonly: 'readonly'
 }
 }, false);
}
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Adding External CSS/Script File

The editor offers you to add external CSS file to style the `<iframe>` element. Easily change the appearance of editor's content using an external CSS file using `styles` field in [IframeSettings](#) property.

Likewise, add the external script file to the `<iframe>` element using `scripts` field of [IframeSettings](#) to provide the additional functionalities to the Rich Text Editor.

## CSHTML

```

@Html.EJS().RichTextEditor("iframe").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).IframeSettings(iframeSettings =>
 iframeSettings.Enable(true).Resources(ViewBag.resources)).Render()

```

**CONTROLLER.CS**

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.resources = new
 {
 styles = new[] { "myStyle.css" }
 };
 return View();
 }
}

```

See Also

- [How to change the editor mode](#)

### ExecCommand in Rich Text Editor

In Rich Text Editor, execCommand used to perform commands for the modification of content in editable area.

The execCommand will perform the following commands.

| Commands          | Description                                                        | Code snippets                                              |
|-------------------|--------------------------------------------------------------------|------------------------------------------------------------|
| ----- ----- ----- |                                                                    |                                                            |
| bold              | Bold the selected content in the Rich Text Editor.                 | <code>rteObj.executeCommand('bold');</code>                |
| italic            | The selected text will be italics.                                 | <code>rteObj.executeCommand('italic');</code>              |
| underline         | Underline the selected text in the Rich Text Editor.               | <code>rteObj.executeCommand('underline');</code>           |
| strikeThrough     | Apply single line strike through formatting for the selected text. | <code>rteObj.executeCommand('strikeThrough');</code>       |
| superscript       | Makes the selected text as superscript (higher).                   | <code>rteObj.executeCommand('superscript');</code>         |
| subscript         | Makes the selected text as subscript (lower).                      | <code>rteObj.executeCommand('subscript');</code>           |
| uppercase         | Change the case of selected text to upper in the content.          | <code>rteObj.executeCommand('uppercase');</code>           |
| lowercase         | Change the case of selected text to lower in the content.          | <code>rteObj.executeCommand('lowercase');</code>           |
| fontColor         | Apply the specified font color for the selected text.              | <code>rteObj.executeCommand('fontColor', 'yellow');</code> |
| fontName          | Apply the specified font name for the selected text.               | <code>rteObj.executeCommand('fontName', 'Arial');</code>   |

| **fontSize** | Apply the specified font size for the selected text. | `rteObj.executeCommand('fontSize', '10pt');` |

| **formatBlock** | Apply the specified format styles for the selected text. | `rteObj.executeCommand('formatBlock', 'H1');` |

| **backColor** | Apply the specified background color the selected text. | `rteObj.executeCommand('backColor', 'red');` |

| **justifyCenter** | Align the content with center margin. | `rteObj.executeCommand('justifyCenter');` |

| **justifyFull** | Align the content with justify margin. | `rteObj.executeCommand('justifyFull');` |

| **justifyLeft** | Align the content with left margin. | `rteObj.executeCommand('justifyLeft');` |

| **justifyRight** | Align the content with right margin. | `rteObj.executeCommand('justifyLeft');` |

| **undo** | Allows to undo the actions. | `rteObj.executeCommand('undo');` |

| **createLink** | Creates a hyperlink to a text or image to a specific location in the content. | `rteObj.executeCommand('createLink', { text: 'Links', url: 'http://', title : 'Link' });` |

| **indent** | Allows to increase the indent level of the content. | `rteObj.executeCommand('indent');` |

| **insertHTML** | Insert the html content to the current cursor position. | `rteObj.executeCommand('insertHTML', 'inserted an html');` |

| **insertOrderedList** | Create a new list item(numbered). | `rteObj.executeCommand('insertOrderedList');` |

| **insertUnorderedList** | Create a new list item(bulleted). | `rteObj.executeCommand('insertUnorderedList');` |

| **outdent** | Allows to decrease the indent level of the content. | `rteObj.executeCommand('outdent');` |

| **redo** | Allows to redo the actions | `rteObj.executeCommand('redo');` |

| **removeFormat** | remove all formatting styles (such as bold, italic, underline, color, superscript, subscript, and more) from currently selected text. | `rteObj.executeCommand('removeFormat');` |

| **insertText** | Insert text to the current cursor position. | `rteObj.executeCommand('insertText', 'inserted a text');` |

| **insertImage** | Insert an image to the current cursor position. | `rteObj.executeCommand('insertImage', { url: 'https://ej2.syncfusion.com/javascript/demos/src/rich-text-editor/images/RTEImage-Feather.png', cssClass: 'rte-img' });` |

| **copyFormatPainter** | Copy the format of selected text and apply it to another text. | `rteObj.executeCommand('copyFormatPainter', formatPainterSettings);` |

| **applyFormatPainter** | Apply the copied format to the selected text. | `rteObj.executeCommand('applyFormatPainter');` |

| **escapeFormatPainter** | Remove the previously copied format and disable the sticky mode | `rteObj.executeCommand('escapeFormatPainter');` |

**Note:** The 'ExecuteCommand' public method is not supported in Syncfusion Markdown Editor

### CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the Rich Text Editor's content

Use the following CSS to customize the default Rich Text Editor's content properties like font-family, font-size and color.

```
`CSS

/ To change font family and font size /
.e-richtexteditor .e-rte-content .e-content,
.e-richtexteditor .e-source-content .e-content {
font-size: 20px;
font-family: Segoe ui;
}

/ To change font color and content background /
.e-richtexteditor .e-rte-content,
.e-richtexteditor .e-source-content {
background: seashell;
color: blue;
}
,
```

#### Customizing the Rich Text Editor's toolbar

Use the following CSS to customize the default color in the Rich Text Editor's toolbar icon.

```
`CSS

/ To change font color for toolbar icon /
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons,
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-icons:active {
color: red;
}

/ To change font color for toolbar button /
.e-toolbar .e-tbar-btn,
.e-toolbar .e-tbar-btn:active,
.e-toolbar .e-tbar-btn:hover {
color: red;
```

```

}
/ To change font color for toolbar button in active state/
.e-richtexteditor .e-rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active .e-icons, .e-richtexteditor .e-
rte-toolbar .e-toolbar-item .e-dropdown-btn.e-active .e-rte-dropdown-btn-text {
color: red;
}
/ To change font color for expanded toolbar items /
.e-richtexteditor .e-rte-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-btn .e-icons,
.e-toolbar.e-extended-toolbar .e-toolbar-extended .e-toolbar-item .e-tbar-btn {
color: red;
}
`

```

### Customizing the Rich Text Editor's character count

Use the following CSS to customize the default color in the Rich Text Editor's character count.

```

`CSS
/ To change font color, font family, font size and opacity /
.e-richtexteditor .e-rte-character-count {
color: red;
font-family: segoe ui;
font-size: 18px;
opacity: 0.54;
padding-bottom: 2px;
padding-right: 14px;
}
`

```

### Keyboard Support

The editor has full keyboard accessibility that includes shortcuts to open and other actions with toolbar items, drop-down lists, and dialogs.

#### HTML Formation Shortcut Key

You can use the following key shortcuts when the Rich Text Editor renders with HTML edit mode.

| Actions | Keyboard shortcuts |

|-----|-----|

| **Toolbar focus | Alt + f10 |**

| **Insert link | Ctrl + k |**

| Insert image | Ctrl + Shift + i |

| Insert audio | Ctrl + Shift + a |

| Insert video | Ctrl + Alt + v |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

| Underline | Ctrl + u |

| Strikethrough | Ctrl + Shift + s |

| Uppercase | Ctrl + Shift + u |

| Lowercase | Ctrl + Shift + l |

| Superscript | Ctrl + Shift + = |

| Subscript | Ctrl + = |

| Indents | Ctrl + ] |

| Outdents | Ctrl + [ |

| HTML source | Ctrl + Shift + h |

| Fullscreen | Ctrl + Shift + f |

| Exit Fullscreen | Esc |

| Justify center | Ctrl + e |

| Justify full | Ctrl + j |

| Justify left | Ctrl + l |

| Justify right | Ctrl + r |

| Clear format | Ctrl + Shift + r |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

| Format Painter Copy | Alt + Shift + c |

| Format Painter Paste | Alt + Shift + v |

## | Format Painter Escape | Esc |

### CSHTML

```
@Html.EJS().RichTextEditor("keyboard").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>).Created("created").Height("500").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).Render()
<script type="text/javascript">
 var rteObj;
 function created() {
 rteObj = document.getElementById("defaultRTE").ej2_instances[0];
 }
 document.addEventListener('keyup', function (e) {
 if (e.altKey && e.keyCode === 84) { /* t */
 // press alt+t to focus the control.
 rteObj.focusIn();
 }
 });
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Bold", "Italic", "Underline",
 "StrikeThrough",
 "FontName", "FontSize", "FontColor", "BackgroundColor",
 "LowerCase", "UpperCase", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList",
 "Outdent", "Indent", "|",

```

```

 "CreateLink", "Image", "CreateTable", "|", "FormatPainter" ,
 "ClearFormat", "Print",
 "SourceCode", "FullScreen", "|", "Undo", "Redo"
 };
 return View();
}
}

```

### Markdown Formation Shortcut Key

You can use the following key shortcuts when the Rich Text Editor renders with Markdown edit mode.

| Actions | Keyboard shortcuts |

|-----|-----|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

| Strikethrough | Ctrl + Shift + s |

| Uppercase | Ctrl + Shift + u |

| Lowercase | Ctrl + Shift + l |

| Superscript | Ctrl + Shift + = |

| Subscript | Ctrl + = |

| Fullscreen | Ctrl + Shift + f |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

### CSHTML

```

@using Syncfusion.EJ2.RichTextEditor
@Html.EJS().RichTextEditor("keyboard").EditorMode(EditorMode.Markdown).Value
((string)ViewBag.value).Created("created").Height(300).ToolbarSettings(e =>
e.Items((object)ViewBag.items)).Render()
<script type="text/javascript">

```



```

var rteObj;
function created() {
 rteObj = this;
}
document.addEventListener('keyup', function (e) {
 if (e.altKey && e.keyCode === 84) { /* t */
 // press alt+t to focus the control.
 rteObj.focusIn();
 }
});
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new[] { "Bold", "Italic", "StrikeThrough", "|",
 "Formats", "OrderedList", "UnorderedList", "|",
 "CreateLink", "Image", "CreateTable", "|", "Undo", "Redo" };
 ViewBag.value = @"The sample is added to showcase **markdown
editing**.
Type or edit the content and apply formatting to view markdown formatted
content.
We can add our own custom formation syntax for the Markdown formation.
The third-party library Marked is used in this sample to convert
markdown into HTML content";
 return View();
 }
}

```

## Custom Key Config

Customize the key config for the keyboard interaction of Rich Text Editor, using the [KeyConfig](#) property.

In the following sample, customize the bold, italic, underline toolbar action with ctrl + alt + b, ctrl + alt + i and ctrl + alt + u respectively.

## CSHTML

```

@Html.EJS().RichTextEditor("customKey").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>

</div>

```

```

<p> Provides HTML view to edit the source directly for
developers.</p>
<p> Supports third - party library integration.</p>
<p> Allows preview of modified content before saving
it.</p>
<p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
<p> Contains undo / redo manager.</p>
<p> Creates bulleted and numbered lists.</p>

</div>).KeyConfig((object)ViewBag.keyConfig).ToolBarSettings(e =>
e.Items((object)ViewBag.items)).Render()

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public KeyModel KeyConfigData = new KeyModel();
 public class KeyModel
 {
 public string bold { get; set; }
 public string italic { get; set; }
 public string underline { get; set; }
 }
 public ActionResult Index()
 {
 KeyConfigData.bold = "ctrl+alt+b";
 KeyConfigData.italic = "ctrl+alt+i";
 KeyConfigData.underline = "ctrl+alt+u";
 ViewBag.keyConfig = KeyConfigData;
 ViewBag.items = new[] { "Bold", "Italic", "Underline",
"StrikeThrough",
"FontName", "FontSize", "FontColor", "BackgroundColor",
"LowerCase", "UpperCase", "|",
"Formats", "Alignments", "OrderedList", "UnorderedList",
"Outdent", "Indent", "|",
"CreateLink", "Image", "CreateTable", "|", "ClearFormat", "Print",
"SourceCode", "FullScreen", "|", "Undo", "Redo" };
 return View();
 }
}

```

## See Also

- [Globalization](#)
- [Accessibility](#)
- [How to customize the saving operation](#)

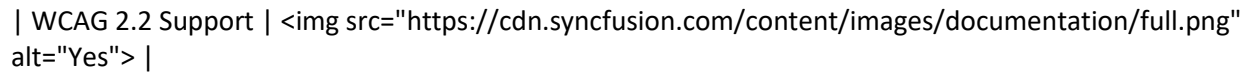
## Accessibility

The Rich Text Editor control has been designed, keeping in mind the WAI-ARIA specifications, and applies the WAI-ARIA roles, states, and properties. This control is characterized by complete ARIA accessibility support that makes it easy for people who use assistive technologies (AT) or those who completely rely on keyboard navigation.

The accessibility compliance for the Rich Text Editor control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| WCAG 2.2 Support |  alt="Yes"> |

| [Section 508 Support](#) |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Intermediate"> |

| Mobile Device Support |  alt="Yes"> |

| [Keyboard Navigation Support](#) |  alt="Yes"> |

| Accessibility Checker Validation |  alt="Intermediate"> |

| Axe-core Accessibility Validation |  alt="Yes"> |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

## ARIA Attributes

The toolbar of Rich Text Editor, assigned the role of Toolbar and has the following list of ARIA attributes.

| Roles and Attributes | Functionalities |

| --- | --- |

| role="toolbar" | This attribute added to the toolbar element describes the actual role of the element.  
|

| aria-orientation | Indicates the toolbar orientation. Default value is horizontal. |

| aria-haspopup | Indicates the popup mode of the toolbar. The default value is false. When popup mode is enabled, attribute value has to be changed to true. |

| aria-disabled | Indicates the disabled state of the toolbar. |

| aria-owns | Identifies an element to define a visual, functional, or contextual parent/child relationship between DOM elements when the DOM hierarchy cannot represent the relationship. In the Rich Text Editor, the attribute contains the ID of the Rich Text Editor to indicate the popup as a child element. |

For further details of toolbar ARIA attributes, refer the accessibility of [Toolbar](#) documentation.

The Rich Text Editor element is assigned the role of application.

| **Roles and Attributes** | **Functionalities** |

| --- | --- |

| role="application" | This attribute added to the Rich Text Editor element describes the actual role of the element. |

| aria-disabled | Indicates the disabled state of the Rich Text Editor. |

## CSHTML

```
@Html.EJS().RichTextEditor("accessibility").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
```

## CONTROLLER.CS

```
public class HomeController : Controller {
```

```

public ActionResult Index () {
 return View ();
}

```

### Keyboard interaction

The Rich Text Editor control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Rich Text Editor control.

#### HTML formation shortcut key

You can use the following key shortcuts when the Rich Text Editor renders with HTML edit mode.

| Actions | Keyboard shortcuts |

|-----|-----|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

| Underline | Ctrl + u |

| Strikethrough | Ctrl + Shift + s |

| Uppercase | Ctrl + Shift + u |

| Lowercase | Ctrl + Shift + l |

| Superscript | Ctrl + Shift + = |

| Subscript | Ctrl + = |

| Indents | Ctrl + ] |

| Outdents | Ctrl + [ |

| HTML source | Ctrl + Shift + h |

| Fullscreen | Ctrl + Shift + f |

| Exit Fullscreen | Esc |

| Justify center | Ctrl + e |

| Justify full | Ctrl + j |

| Justify left | Ctrl + l |

| Justify right | Ctrl + r |

| Clear format | Ctrl + Shift + r |

| Ordered list | Ctrl + Shift + o |

| Unordered list | Ctrl + Alt + o |

| Format Painter Copy | Alt + Shift + c |

| Format Painter Paste | Alt + Shift + v |

| Format Painter Escape | Esc |

#### *Markdown formation shortcut key*

You can use the following key shortcuts when the Rich Text Editor renders with Markdown edit mode

| Actions | Keyboard shortcuts |

|-----|-----|

| Toolbar focus | Alt + f10 |

| Insert link | Ctrl + k |

| Insert image | Ctrl + Shift + i |

| Insert table | Ctrl + Shift + e |

| Undo | Ctrl + z |

| Redo | Ctrl + y |

| Copy | Ctrl + c |

| Cut | Ctrl + x |

| Paste | Ctrl + v |

| Bold | Ctrl + b |

| Italic | Ctrl + i |

| Strikethrough | Ctrl + Shift + s |

| Uppercase | Ctrl + Shift + u |

| Lowercase | Ctrl + Shift + l |

| Superscript | Ctrl + Shift + = |

| Subscript | Ctrl + = |

| Fullscreen | Ctrl + Shift + f |

| Ordered list | Ctrl + Shift + o |

## | Unordered list| Ctrl + Alt + o |

### Ensuring accessibility

The Rich Text Editor control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Rich Text Editor control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Rich Text Editor control with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC controls](#)

### How To

#### Add Google fonts

To use web fonts in Rich Text Editor, it is not needed for the web fonts to be present in local machine.

To add the web fonts to Rich Text Editor, we need to refer the web font links and add the font names in the [FontFamily](#) property.

#### CSHTML

```
@Html.EJS().RichTextEditor("google-font").FontFamily(e =>
e.Items((object)ViewBag.fontItems)).ToolbarSettings(e =>
e.Items((object)ViewBag.tools)).ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto">
<link rel="stylesheet"
href="http://fonts.googleapis.com/css?family=Great+Vibes">
```

**CONTROLLER.CS**

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 object item1 = new
 {
 text = "Segoe UI",
 value = "Segoe UI"
 };
 object item2 = new
 {
 text = "Roboto",
 value = "Roboto"
 };
 object item3 = new
 {
 text = "Great vibes",
 value = "Great Vibes,cursive"
 };
 object item4 = new
 {
 text = "Noto Sans",
 value = "Noto Sans"
 };
 object item5 = new
 {
 text = "Impact",
 value = "Impact,Charcoal,sans-serif"
 };
 object item6 = new
 {
 text = "Tahoma",
 value = "Tahoma,Geneva,sans-serif"
 };
 ViewBag.fontItems = new[] { item1, item2, item3, item4, item5, item6 };

 ViewBag.tools = new[] {
 "Bold", "Italic", "Underline", "StrikeThrough",
 "FontName", "FontSize", "FontColor", "BackgroundColor",
 "LowerCase", "UpperCase", "|",
 "Formats", "Alignments", "OrderedList", "UnorderedList",
 "Outdent", "Indent", "|",
 "CreateLink", "Image", "CreateTable", "|", "ClearFormat", "Print",
 "SourceCode", "FullScreen", "|", "Undo", "Redo"
 };

 return View();
 }
}

```

The below font style links are referred in the page.

`typescript

<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Roboto">



```
<link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Great+Vibes">
```

In the above sample, you can see that we have added two Google web fonts (**Roboto** and **Great vibes**) to Rich Text Editor.

### Customize the placeholder style

By using **e-rte-placeholder** class, you can customize the placeholder style.

#### CSHTML

```
@Html.EJS().RichTextEditor("placeholder").Placeholder("Type
Something").Render()
<style>
 .e-richtexteditor .e-rte-placeholder {
 font-family: monospace;
 color: deeppink;
 }
</style>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Restrict the image uploading while uploading with large size

By using the Rich text editor's **ImageUploading** event, you can get the image size before uploading and restrict the image to upload, when the given image size is greater than the allowed size.

In the following, we have validated the image size before uploading and determined whether the image has been uploaded or not.

#### CSHTML

```
@Html.EJS().RichTextEditor("editor").InsertImageSettings(obj =>
obj.SaveUrl("https://aspnetmvc.syncfusion.com/services/api/uploadbox/Save").
Path("../Images/")).ImageUploading("onImageUploading").Render()
<script>
 function onImageUploading(args) {
 console.log("file is uploading");
 var imgSize = 500000;
 var sizeInBytes = args.fileData.size;
 if (imgSize < sizeInBytes) {
 args.cancel = true;
 }
 }
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Save Rich Text Editor content in a file in the server

Rich Text Editor content can be passed from view to controller through `XMLHttpRequest` post. Content will be sent to the corresponding method into the controller and this value can be saved in a text file or any other format using `StreamWriter`. Refer to the following given code.

### CSHTML

```
<div>
 <div>
 @Html.EJS().RichTextEditor("saveFile").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is
 what you get') editor that provides the best user experience to create and
 update the content.
 Users can format their content using standard toolbar
 commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV >
 modes </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
 functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
 developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
 it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks,
 uploads, etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

 </div>).Render()
 </div>
 <div> <button id="btn">save content</button></div>
</div>
<script>
 document.getElementById('btn').addEventListener('click', function () {
 var obj = document.getElementById('saveFile').ej2_instances[0];
 var value = JSON.stringify({ text: obj.value });
 const Http = new XMLHttpRequest();
 const url = '@Url.Action("Save")';
 Http.open("POST", url);
 Http.setRequestHeader('Content-Type', 'application/json');
 Http.send(value);
 });
</script>
```

```
});
</script>
```

### SAVEFILE.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
 public class RichTextEditorValue
 {
 public string text { get; set; }
 }
 [HttpPost]
 public ActionResult
Save([System.Web.Http.FromBody]RichTextEditorValue value)
 {
 string RootPath = Server.MapPath("~/data.txt");
 StreamWriter writeFile = new StreamWriter(RootPath, true);
 writeFile.WriteLine(value.text);
 writeFile.Close();
 writeFile.Dispose();
 return View();
 }
}
```

### Set the cursor at the specific range

This can be achieved by using `setRange` method in the Rich Text Editor using `NodeSelection` instance. In this below sample, we have passed the text node (specific location in Rich Text Editor content) in `setStart` method and passed the range in `setRange` method of Rich Text Editor.

### CSHTML

```
@Html.EJS().RichTextEditor("cursor").ContentTemplate(@<div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
 get') editor that provides the best user experience to create and update the
 content.
 Users can format their content using standard toolbar commands.
 </p>
 <p><b id='key'> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
 </p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
 functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
</div>
```

```

 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
<input id="btn" type="button" value="Set cursor point" onclick="btnClick()"
/>
<script>
 function btnClick() {
 var selectioncursor = new ej.richtexteditor.NodeSelection();
 var element =
document.getElementById("cursor").ej2_instances[0].contentModule.getDocument
().getElementById("key");
 var range = document.createRange();
 range.setStart(element, 1); // to set the range
 selectioncursor.setRange(document, range); // to set the cursor
 }
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Capture ctrl+s to update the value

To achieve this, we need to bind the `keydown` event to the Rich Text Editor content and capture the `ctrl + s` key press using its `keyCode`.

In the `keydown` event handler, the `updateValue` method is called to update the `Value` property and then we can save the content in the required database using the same.

## CSHTML

```

@Html.EJS().RichTextEditor("default").Created("onCreate").ContentTemplate(@<
div>
 <p>
 The Rich Text Editor control is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
 Users can format their content using standard toolbar commands.
 </p>
 <p> Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
</p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>

</div>

```

```

 <p> Provides HTML view to edit the source directly for
 developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
 it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
 etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>).Render()
<script>
 function onCreate() {
 var rteObj = this;
 rteObj.contentModule.getDocument().addEventListener("keydown",
function (e) {
 if (e.key === 's' && e.ctrlKey === true) {
 e.preventDefault(); // to prevent default ctrl+s action
 rteObj.updateValue(); // to update the value after editing
 var value = rteObj.value; // you can get the Rich Text
 Editor content to save in the desired database
 }
});
 }
}
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Render the RichTextEditorFor control

The RichTextEditorFor control can be rendered by passing values from the controller. The formatted Rich Text Editor value is retrieved when submitting the form using the post method.

In the following sample, the RichTextEditorFor control is rendered.

### CSHTML

```

@model EJ2MVCSampleBrowser.Controllers.RichTextEditorModel
@using (Html.BeginForm()) {
 @Html.ValidationSummary(true)
 <div class="col-lg-12 control-section">
 <div style='padding-top:40px;'>
 <div id="wrapper">
 @Html.EJS().RichTextEditorFor(model => model.Value).Render()
 <div id="errorMessage">
 @Html.ValidationMessageFor(model => model.Value)
 </div>
 <div id="submitbutton">
 @Html.EJS().Button("btn").Content("Submit").Render()
 </div>
 </div>
 </div>
 </div>
}

```

```

 </div>
 </div>
</div>
}
<style>
 #submitButton {
 margin: 10px auto;
 text-align: center;
 }
 #errorMessage {
 color: red;
 }
</style>

```

### CONTROLLER.CS

```

using System.ComponentModel.DataAnnotations;
public class RichTextEditorModel
{
 [Required(ErrorMessage = "Value is required")]
 // Specify AllowHtml attribute on MVC application alone
 [AllowHtml]
 public string Value { get; set; }
}
public partial class HomeController : Controller
{
 RichTextEditorModel rteModel = new RichTextEditorModel();
 public ActionResult RichTextEditorFor()
 {
 rteModel.Value = "<p>Type or edit the content to post the
Rich Text Editor value.</p>";
 return View(rteModel);
 }
 [HttpPost]
 public ActionResult RichTextEditorFor(RichTextEditorModel model)
 {
 rteModel.Value = model.Value;
 return View(rteModel);
 }
}

```

The output will be as follows.

The screenshot displays a web form with a Rich Text Editor. The toolbar includes icons for Bold (B), Italic (I), Underline (U), Paragraph, Bulleted List, Numbered List, Link, and Unlink. The text area contains the placeholder text: "Type or edit the content to post the Rich Text Editor value." Below the text area is a "SUBMIT" button.

Rename uploaded images in server before inserting it in the Rich Text Editor

By using the [InsertImageSettings](#) property, you can specify the server handler to upload the selected image. Then you can bind the [ImageUploadSuccess](#) event, to receive the modified file name from the server and update it in the Rich Text Editor's insert image dialog.

Refer `rename.cs` controller file for configure the server-side.

**Note:** The runnable demo application is available in this [Github](#) repository.

### CSHTML

```
@Html.EJS().RichTextEditor("editor").InsertImageSettings(obj =>
obj.SaveUrl("/Home/Rename").Path("../Uploads/")).ImageUploadSuccess("onImage
UploadSuccess").Render()
<script>
 function onImageUploadSuccess(args) {
 if (args.e.currentTarget.getResponseHeader('name') != null) {
 args.file.name = args.e.currentTarget.getResponseHeader('name');
 var filename = document.querySelectorAll(".e-file-name")[0];
 filename.innerHTML =
args.file.name.replace(document.querySelectorAll(".e-file-
type")[0].innerHTML, '');
 filename.title = args.file.name;
 }
 }
</script>
```

### RENAME.CS

```
using System;
using System.IO;
using System.Web;
using System.Web.Mvc;
namespace RenameImage.Controllers
{
 public class HomeController : Controller
 {
 int x = 0;
 string imageFile;
 public ActionResult Index()
 {
 return View();
 }
 [AcceptVerbs("Post")]
 public void Rename(HttpPostedFileBase UploadFiles)
 {
 try
 {
 if (UploadFiles != null)
 {
 imageFile = UploadFiles.FileName;
 if (UploadFiles != null)
 {
 var fileSave =
System.Web.HttpContext.Current.Server.MapPath("~/Uploads");
 // Create a new directory, if it does not exists
 }
 }
 }
 }
 }
}
```

```

 if (!System.IO.Directory.Exists(fileSave))
 {
 System.IO.Directory.CreateDirectory(fileSave);
 }
 var fileName =
Path.GetFileName(UploadFiles.FileName);
 var fileSavePath = Path.Combine(fileSave, fileName);
// Rename a uploaded image file name
 while (System.IO.File.Exists(fileSavePath))
 {
 imageFile = "rteImage" + x + "-" + fileName;
 fileSavePath = Path.Combine(fileSave,
imageFile);

 x++;
 }
 if (!System.IO.File.Exists(fileSavePath))
 {
 UploadFiles.SaveAs(fileSavePath);
// Modified file name shared through response
header by adding custom header
 Response.Headers.Add("name", imageFile);
 Response.StatusCode = 200;
 Response.ContentType = "application/json";
charset=utf-8";
 }
 }
}
catch (Exception)
{
 Response.StatusCode = 204;
}
}
}
}

```

## File Attachments

The Rich Text Editor allows you to attach a file based on the file upload. You can attach your files using the file upload or drag-and-drop from your local path. When the file upload gets success, the attachment link inserts into the content.

In the below sample, configure the `saveUrl` and `path` properties to achieve file attachments.

1. `saveUrl`: Provides service URL to save the files.
2. `path`: Specifies the location to store the image.

The following sample illustrates how to attach a file in the Rich Text Editor.

## CSHTML

```
@Html.EJS().RichTextEditor("defaultRTE").ContentTemplate(@<div>
 <p>
 The Rich Text Editor component is WYSIWYG ('what you see is what you
get') editor that provides the best user experience to create and update the
content.
```



```

 Users can format their content using standard toolbar commands.
 </p>
 <p>Key features:</p>

 <p>Provides < IFRAME > and < DIV > modes
 </p>
 <p>Capable of handling markdown editing.</p>
 <p>Contains a modular library to load the necessary
functionality on demand.</p>
 <p>Provides a fully customizable toolbar.</p>
 <p>Provides HTML view to edit the source directly for
developers.</p>
 <p>Supports third - party library integration.</p>
 <p>Allows preview of modified content before saving
it.</p>
 <p>Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p>Contains undo / redo manager.</p>
 <p>Creates bulleted and numbered lists.</p>

</div>
).InsertImageSettings(obj =>
obj.SaveUrl("[SERVICE_HOSTED_PATH]/api/uploadbox/Save").Path("../Files/").C
reated("created").Render()
@Html.EJS().Uploader("UploadFiles").DropArea(".e-
richtexteditor").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
@Url.Content("[SERVICE_HOSTED_PATH]/api/uploadbox/Save") }).Success("onImageU
ploadSuccess").Render()
<script>
 var rteObj;
 var selection = new ej.richtexteditor.NodeSelection();
 var range;
 let saveSelection;
 function created() {
 rteObj = this;
 }
 function onImageUploadSuccess(e) {
 rteObj.contentModule.getEditPanel().focus();
 range = selection.getRange(document);
 saveSelection = selection.save(range, document);
 var fileUrl = document.URL + rteObj.insertImageSettings.path +
e.file.name;
 if (rteObj.formatter.getUndoRedoStack().length === 0) {
 rteObj.formatter.saveData();
 }
 saveSelection.restore();
 rteObj.executeCommand('createLink', { url: fileUrl, text: fileUrl,
selection: saveSelection });
 rteObj.formatter.saveData();
 rteObj.formatter.enableUndo(rteObj);
 this.clearAll();
 }
</script>

```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

To config server-side handler, refer the below code.

```
`csharp
int x = 0;
string file;
[AcceptVerbs("Post")]
public void Save()
{
 try
 {
 var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
 file = httpPostedFile.FileName;
 if (httpPostedFile != null)
 {
 Console.WriteLine(System.Web.HttpContext.Current.Server.MapPath("~/Files"));
 var fileSave = System.Web.HttpContext.Current.Server.MapPath("~/Files");
 if (!Directory.Exists(fileSave))
 {
 Directory.CreateDirectory(fileSave);
 }
 var fileName = Path.GetFileName(httpPostedFile.FileName);
 var fileSavePath = Path.Combine(fileSave, fileName);
 while (System.IO.File.Exists(fileSavePath))
 {
 file = "rte" + x + "-" + fileName;
 fileSavePath = Path.Combine(fileSave, file);
 x++;
 }
 if (!System.IO.File.Exists(fileSavePath))
```

```

{
 httpPostedFile.SaveAs(fileSavePath);
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.Headers.Add("name", file);
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File uploaded succesfully";
 Response.Headers.Add("url", fileSavePath);
 Response.End();
}
}
}
catch (Exception e)
{
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 204;
 Response.Status = "204 No Content";
 Response.StatusDescription = e.Message;
 Response.End();
}
}
,

```

#### Format code block using toolbar button

You can configure code block formatting as a separate toolbar button by adding the **InsertCode** keyword within the [ToolbarSettings](#) items property.

The InsertCode button has a toggle state to apply code block formatting to the editor and remove code block formatting from the editor.

The following sample demonstrates how to config the InsertCode button in toolbar and set the background color to “pre” tag for highlighting the code block.

#### CSHTML

```
@Html.EJS().RichTextEditor("code").ContentTemplate(@<div>
 <p>
```

The Rich Text Editor control is WYSIWYG ('what you see is what you get') editor that provides the best user experience to create and update the content.

Users can format their content using standard toolbar commands.

```

</p>
<p>Key features:</p>

 <p> Provides < IFRAME > and < DIV > modes
</p>
 <p> Capable of handling markdown editing.</p>
 <p> Contains a modular library to load the necessary
functionality on demand.</p>
 <p> Provides a fully customizable toolbar.</p>
 <p> Provides HTML view to edit the source directly for
developers.</p>
 <p> Supports third - party library integration.</p>
 <p> Allows preview of modified content before saving
it.</p>
 <p> Handles images, hyperlinks, video, hyperlinks, uploads,
etc.</p>
 <p> Contains undo / redo manager.</p>
 <p> Creates bulleted and numbered lists.</p>

</div>
).Height("500").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).Render()
<style>
 .e-richtexteditor .e-rte-content .e-content pre {
 padding: 10px;
 background: #F4F5F7;
 }
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.items = new object[] { "InsertCode" };
 return View();
 }
}

```

The output will be as follows.

<>

The rich text editor component is WYSIWYG ("what you see is what you get") editor that provides the best user experience to create and update the content.

Users can format their content using standard toolbar commands.

**Key features:**

- Provides <IFRAME> and <DIV> modes
- Capable of handling markdown editing.
- Contains a modular library to load the necessary functionality on demand.
- Provides a fully customizable toolbar.
- Provides HTML view to edit the source directly for developers.
- Supports third-party library integration.
- Allows preview of modified content before saving it.
- Handles images, hyperlinks, video, hyperlinks, uploads, etc.
- Contains undo/redo manager.
- Creates bulleted and numbered lists.

## Migration from Essential JS 1

This article describes the API migration process of Rich Text Editor control from Essential JS 1 to Essential JS 2.

### Accessibility

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Keyboard Navigation | **Property:** AllowKeyboardNavigation <br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").AllowKeyboardNavigation(true).Render();} | No separate Property for enable/disable keyboard navigation. Its enabled by default. |

| Localization | **Property:** Locale <br/> <br/> @{Html.EJ().RichTextEditor("rteSample").Locale("en-US").Render();} | **Property:** Locale <br/> <br/> @{Html.EJS().RichTextEditor("default").Locale("en-US").Render()} |

| RTL | **Property:** EnableRtl<br/> <br/>

@{Html.EJ().RichTextEditor("rteSample").EnableRTL(true).Render();} | **Property:** EnableRtl <br/> <br/>@{Html.EJS().RichTextEditor("rtl").EnableRtl(true).Render()} |

| Key Config | Not Applicable | **Property:** KeyConfig<br/>

<br/>@{Html.EJS().RichTextEditor("default").KeyConfig((object)ViewBag.keyConfig).Render()} |

| Tab Key Navigation | **Property:** EnableTabKeyNavigation <br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").EnableTabKeyNavigation(true).Render();} |

**Property:** EnableTabKey<br/>

<br/>@{Html.EJS().RichTextEditor("default").EnableTabKey(true).Render()} |

| Key Down | **Event:** KeyDown<br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt => evt.KeyDown("onkeydown")).Render();} | Not Applicable |

| Key Up | **Event:** keyUp<br/> <br/>@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt => evt.keyUp("keyup")).Render();} | Not Applicable |

## Toolbar

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| showToolbar | **Property:** ShowToolbar<br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ShowToolbar(true).Render();} | **Property:** Enable  
<br/>

<br/>@Html.EJS().RichTextEditor("types").Value((string)ViewBag.value).Height("720px").Toolba  
rSettings(e => e.Enable(true)).Render() |

| Tools item | **Property:** Tools <br/> <br/>List<String> lists = new List<string>() { "unorderedList",  
"orderedList" }; <br/> <br/>@{Html.EJ().RichTextEditor("rteSample").Tools(tool =>  
tool.Lists(lists)).Render();} | **Property:** Items<br/>

<br/>@Html.EJS().RichTextEditor("customtool").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).Render() |

| Tools-list | **Property:** ToolsList<br/> <br/> @{ List<String> toolsList = new List<string>() { "style",  
"lists", "doAction", "links", "images" };<br/> <br/>

@{Html.EJ().RichTextEditor("rteSample").ToolsList(toolsList).Render();} | **Property:** Items<br/>  
<br/>@Html.EJS().RichTextEditor("customtool").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).Render() |

| Toolbar Overflow Mode | **Property:** ToolbarOverflowMode <br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ToolbarOverflowMode(ToolbarOverflowMode.In  
line).Render();} | **Property:** Type <br/> <br/>1.Expand<br/> <br/> 2.MultiRow<br/>

<br/>@Html.EJS().RichTextEditor("types").ToolbarSettings(e =>  
e.Type(Syncfusion.EJ2.RichTextEditor.ToolbarType.Expand)).Render() |

| Floating Toolbar | Not Applicable | **Property:** EnableFloating <br/>

<br/>@Html.EJS().RichTextEditor("types").ToolbarSettings(e =>e.EnableFloating(true)).Render()  
|

| Floating Toolbar Offset | Not Applicable | **Property:** FloatingToolbarOffset <br/>

<br/>@Html.EJS().RichTextEditor("types").ToolbarSettings(e  
=>e.EnableFloating(true)).FloatingToolbarOffset(50).Render() |

| Inline toolbar | Not Applicable | **Property:** InlineMode<br/> <br/>

@Html.EJS().RichTextEditor("defaultRTE").Value((string)ViewBag.value).InlineMode(e=>e.Enabl  
e(true).OnSelection(true)).ToolbarSettings(e => e.Items((object)ViewBag.items)).Render() |

| Quick Toolbar | Not Applicable | **Property:** Quicktoolbarsettings<br/> <br/>1.image<br/>

<br/>2.link<br/> <br/>3.table<br/> <br/>4.clipboard<br/> <br/>  
@Html.EJS().RichTextEditor("inline").ToolbarSettings(e =>

```
e.Items((object)ViewBag.items)).QuickToolbarSettings(e
=>e.Image((object)ViewBag.image)).Format(t=>t.Width("40px")).Value((string)ViewBag.value).Render() |
```

```
| Enable Toolbar Item | Method: enableToolbarItem()

var rteObj = $("#rte ")
.data("ejRTE ");

rteObj.enableToolbarItem() | Method: enableToolbarItem()

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.enableToolbarItem(); |
```

```
| Disable Toolbar Item | Method: disableToolbarItem()

 var rteObj = $("#rte
").data("ejRTE ");

rteObj.disableToolbarItem() | Method: disableToolbarItem()

var rteObj = document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.disableToolbarItem(); |
```

```
| Remove Toolbar Item | Method: removeToolbarItem()

var rteObj = $("#rte
").data("ejRTE ");

 rteObj.removeToolbarItem() | Method: removeToolbarItem()

 var rteObj = document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.removeToolbarItem() |
```

```
| Toolbar Click | Not Applicable | Event: ToolbarClick

@Html.EJS().RichTextEditor("defaultRTE").ToolbarClick("onClick").Render() |
```

```
| Show Full Screen | Not Applicable | Method: showFullScreen()

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

 rteObj.
showFullScreen() |
```

### Custom Formats and Fonts

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

```
|-----|-----|-----|
```

```
| Format | Property: Format

@{Html.EJ().RichTextEditor("rteSample").Format(format).Render();} | Property: Format


```

```

@Html.EJS().RichTextEditor("defaultRTE").Format(t=>t.Width("70px").Default("Paragraph
").Types("items")).ToolbarSettings(e => e.Items((object)ViewBag.items)).Render() |
```

```
| Font Size | Property: FontSize

@{Html.EJ().RichTextEditor("rteSample").FontSize(size).Render();} | Property: FontSize

@Html.EJS().RichTextEditor("defaultRTE").FontSize(t=>t.Width("70px").Default("Paragrap
h").Types("items")).ToolbarSettings(e => e.Items((object)ViewBag.items)).Render() |
```

```
| Font Family | Property: FontName

@{Html.EJ().RichTextEditor("rteSample").FontName(fontname).Render();} | Property:
FontFamily

@Html.EJS().RichTextEditor("defaultRTE").FontFamily(t=>t.Width("70px").Default("Paragr
aph").Types("items")).ToolbarSettings(e => e.Items((object)ViewBag.items)).Render() |
```

```
| Show Font Option | Property: ShowFontOption

@{Html.EJ().RichTextEditor("rteSample").ShowFontOption(true).Render();} | Property:
FontFamily

viewBag.fontFamily = new {default= "0", types = "items", width = "70px"}
```

<br/>

<br/>@Html.EJS().RichTextEditor("defaultRTE").FontFamily(f=>e.Items((object)ViewBag.fontFamily)).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render() |

### Custom Font Colors

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Font Color | **Property:** ColorCode <br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ColorCode(colorCode).Render();} | **Property:** FontColor<br/>

<br/>@Html.EJS().RichTextEditor("defaultRTE").FontColor(t=>t.columns("10").default("#fff").modeSwitcher(false)).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render() |

| Background Color | Not Applicable | **Property:** BackgroundColor<br/>

<br/>@Html.EJS().RichTextEditor("defaultRTE").BackgroundColor(t=>t.columns("10").Default("#fff").ModeSwitcher(false).Mode("Palette")).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render() |

| Color Palette Columns | **Property:** ColorPaletteColumns <br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ColorPaletteColumns(10).Render();} | **Property:** BackgroundColor<br/>

<br/>@Html.EJS().RichTextEditor("defaultRTE").BackgroundColor(b=>b.Columns(10)).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render() |

| Color Palette Rows | **Property:** ColorPaletteRows <br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ColorPaletteRows(6).Render();} | Not Applicable |

| Color Picker Type | **Method:** setColorPickerType () <br/> <br/> var rteObj = \$("#rte ").data("ejRTE"); <br/> <br/> rteObj.setColorPickerType("picker") | **Property:** ColorModeType <br/> <br/>

@Html.EJS().RichTextEditor("defaultRTE").BackgroundColor(b=>b.Mode("Palette")).ToolBarSettings(e => e.Items((object)ViewBag.items)).Render() |

### Link

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Link | Not Applicable | **Property:** CreateLink<br/> <br/> ViewBag.item = new[] { "CreateLink" }<br/>

<br/>@Html.EJS().RichTextEditor("defaultRTE").ToolBarSettings(e => e.Items((object)ViewBag.items)).Render() |

| Quick Toolbar | Not Applicable | **Property:** Link<br/> <br/> ViewBag.link = new [] { Open, Edit, UnLink }<br/> <br/>@Html.EJS().RichTextEditor("defaultRTE").ToolBarSettings(e =>

e.Items((object)ViewBag.items)).QuickToolBarSettings(e => e.Link((object)ViewBag.link)).Render() |

### Image

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|



| Image | Not Applicable | **Property:** Image <br/> <br/> viewBag.item = new [] { "Image" } <br/>  
<br/>@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).Render() |

| Quick Toolbar | Not Applicable | **Property:** Image <br/> <br/>viewBag.link = new [] { "Replace",  
"Align", "Caption", "Remove", "-", "InsertLink", "OpenImageLink", "EditImageLink",  
"RemoveImageLink", "Display", "AltText", "Dimension" } <br/>  
<br/>@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).QuickToolbarSettings(e =>  
e.Image((object)ViewBag.image)).Render() |

| Image Setting | **Property:** Images<br/> <br/> List<String> images = new List<string>() { "image" };  
<br/> <br/>@{Html.EJ().RichTextEditor("rteSample").Tools(tool=>tool.Image(image)).Render();}  
| **Property:** InsrtlImageSettings <br/> <br/>@Html.EJS().RichTextEditor("image").ToolbarSettings(e  
=>  
e.Items((object)ViewBag.items)).InsertImageSettings(obj=>obj.Display("inline")).Value((string)V  
iewBag.value).Render() |

#### Table

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Table | Not Applicable | **Property:** CreateTable<br/> <br/> viewBag.item = new [] { "CreateTable" }  
<br/> <br/>@Html.EJS().RichTextEditor("table").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).Render() |

| Quick Toolbar | Not Applicable | **Property:** Table<br/> <br/>viewBag.table = new []  
{ "TableHeader", "TableRows", "TableColumns", "BackgroundColor", "-", "TableRemove",  
"Alignments", "TableCellVerticalAlign", "Styles"  
}<br/><br/>@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).QuickToolbarSettings(e =>  
e.Table((object)ViewBag.table)).Render() |

| Table Setting | **Property:** TableColumns, TableRows<br/>  
<br/>@{Html.EJ().RichTextEditor("rteSample").TableColumns(10).Render();} <br/>  
<br/>@{Html.EJ().RichTextEditor("rteSample").TableRows(10).Render();} | **Property:**  
TableSettings<br/> <br/>@Html.EJS().RichTextEditor("image").ToolbarSettings(e =>  
e.Items((object)ViewBag.items)).TableSettings(obj=>obj.width("80%")).Render() |

| Custom table | **Property:** ShowCustomTable<br/>  
<br/>@{Html.EJ().RichTextEditor("rteSample").ShowCustomTable(true).Render();} | Not  
Applicable |

| Insert Column | **Method:** insertColumn([before],[cell])<br/> <br/> var rteObj = \$("#rte  
").data("ejRTE "); <br/> <br/>rteObj.insertColumn(true, "<td></td>") | **Property:** TableColumns  
<br/> <br/> viewBag.table = new [] { "TableColumns" } <br/>  
<br/>@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>

```
e.Items((object)ViewBag.items)).QuickToolbarSettings(e =>
e.Table((object)ViewBag.table)).Render() |
```

```
| Insert Row | Method: insertRow([before],[cell])

 var rteObj = $("#rte").data("ejRTE");

 rteObj.insertRow(true, "<tr></tr>") | Property:

 TableRows ViewBag.table =
new [] { "TableRows" }

 @Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).QuickToolbarSettings(e =>
e.Table((object)ViewBag.table)).Render() |
```

```
| Remove Table | Method: removeTable([table])

 var rteObj = $("#rte").data("ejRTE");

 rteObj.removeTable("table") | Property: TableRemove

 ViewBag.table =
new [] { "TableRemove" }

 @Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e
=> e.Items((object)ViewBag.items)).QuickToolbarSettings(e =>
e.Table((object)ViewBag.table)).Render() |
```

### Counts

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Character Count | **Property:**

```
ShowCharCount

@{Html.EJ().RichTextEditor("rteSample").ShowCharCount(true).Rende
r();} | Property: ShowCharCount

```

```

@Html.EJS().RichTextEditor("defaultRTE").ToolbarSettings(e =>
e.Items((object)ViewBag.items)).ShowCharCount(true).Render() |
```

| Word Count | **Property:** ShowWordCount<br/>

```

@{Html.EJ().RichTextEditor("rteSample").ShowWordCount(true).Render();} | Not Applicable
|
```

| Maximum Length | **Property:** MaxLength <br/>

```

@{Html.EJ().RichTextEditor("rteSample").MaxLength(1000).Render();} | Property:
MaxLength

@Html.EJS().RichTextEditor("defaultRTE").MaxLength(1000).Render() |
```

### IFrame

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Enable Iframe | By Default, enabled | **Property:** IframeSettings<br/>

```

@Html.EJS().RichTextEditor("iframe").Value((string)ViewBag.value).IframeSettings(iframeS
ettings => iframeSettings.Enable(true)).Render() |
```

| Attributes | **Property:** IframeAttributes<br/>

```

@{Html.EJ().RichTextEditor("rteSample").Width("800px").IframeAttributes(new
Dictionary<string, object> { { "style", "background-color:#e0ffff color:#6495ed;" } }).Render();} |
```

**Property:** Attributes <br/>

```

@Html.EJS().RichTextEditor("iframe").Value((string)ViewBag.value).Height("500").IframeS
ettings(iframeSettings => iframeSettings.Enable(true)).Render()

<script>var
iframeRTE; function created() {

 iframeRTE = this;

this.setProperties({

```

```

</>iframeSettings: { enable: true, attributes: { readonly: "readonly" } }, false);
}</></></script> |
| Resource | Property: external-CSS</>
</>@{Html.EJ().RichTextEditor("rteSample").ExternalCSS("/Content/Css/iframe-
custom.css").Render();} | Property: Attributes</>
</>@Html.EJS().RichTextEditor("iframe").Value((string)ViewBag.value).Height("500").IframeS
ettings(iframeSettings => iframeSettings.Enable(true)).Render()</> </></script></> </>
var iframeRTE;</> </>function created() {</> </>iframeRTE = this;</>
</>this.setProperties({ iframeSettings: { enable: true, resources: { styles: ["myStyle.css"]} },
false);}</> </></script> |

```

### Editor Mode

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Editor Mode | Not Applicable | **Property:** EditorMode </> </>1.HTML</> </>2.Markdown  
</>

```

</>@Html.EJS().RichTextEditor("types").Value((string)ViewBag.value).EditorMode(EditorMod
e.Html).Render() |

```

### Undo

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Undo Stack Limit | **Property:** UndoStackLimit </>

```

</>@{Html.EJ().RichTextEditor("rteSample").UndoStackLimit(50).Render();} | Property:
UndoRedoSteps </>

```

```

</>@Html.EJS().RichTextEditor("types").Value((string)ViewBag.value).UndoRedoSteps(30).Re
nder() |

```

| Undo Redo Timer | Not Applicable | **Property:** UndoRedoTimer</>

```

</>@Html.EJS().RichTextEditor("types").Value((string)ViewBag.value).UndoRedoTimer(300).R
ender() |

```

### Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

|-----|-----|-----|

| Allow Editing | **Property:** AllowEditing </>

```

</>@{Html.EJ().RichTextEditor("rteSample").AllowEditing(true).Render();} | Property:
Readonly</> </>@Html.EJS().RichTextEditor("default").Readonly(true).Render() |

```

| Auto Focus | **Property:** AutoFocus </>

```

</>@{Html.EJ().RichTextEditor("rteSample").AutoFocus(true).Render();} | Not Applicable |

```

| Auto Height | **Property:** AutoHeight </>

```

</>@{Html.EJ().RichTextEditor("rteSample").AutoHeight(true).Render();} | Not Applicable |

```

| Paste Clean Up | **Property:** paste-cleanup-settings  
 <br/><br/>@Html.EJ().RichTextEditor("rteSample").PasteCleanupSettings(c=>c.CleanCSS(true).CleanElements(true).ListConversion(true).RemoveStyles(true)).Render();} | Not Applicable |

| Css Class | **Property:** CssClass <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").CssClass("customstyles").Render();} | **Property:** CssClass <br/> <br/>@Html.EJS().RichTextEditor("default").CssClass("customStyle").Render() |

| Enabled | **Property:** Enabled <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").Enabled(true).Render();} | **Property:** Enabled<br/>  
 <br/>@Html.EJS().RichTextEditor("default").Enabled(true).Render() |

| Html Encode | **Property:** EnableHtmlEncode <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").EnableHtmlEncode(true).Render();} | **Property:** EnableHtmlEncode<br/>  
 <br/>@Html.EJS().RichTextEditor("default").EnableHtmlEncode(true).Render() |

| Persistence | **Property:** EnablePersistence <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").EnablePersistence(true).Render();} | **Property:** EnablePersistence<br/>  
 <br/>@Html.EJS().RichTextEditor("default").EnablePersistence(true).Render() |

| Resize | **Property:** EnableResize <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").EnableResize(true).Render();} | Not Applicable |

| XHTML | **Property:** EnableXHTML <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").EnableXHTML(true).Render();} | Not Applicable |

| Height | **Property:** Height <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").Height("350px").Render();} | **Property:** Height<br/> <br/>@Html.EJS().RichTextEditor("default").Height("300px").Render() |

| Width | **Property:** Width <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").Width("500px").Render();} | **Property:** Width  
 <br/> <br/>@Html.EJS().RichTextEditor("default").Width("500px").Render() |

| Html Attributes | **Property:** HtmlAttributes <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").Width("800px").HtmlAttributes(htmlattr).Render();} | **Property:** HtmlAttributes<br/>  
 <br/>@Html.EJS().RichTextEditor("default").HtmlAttributes((object)ViewBag.attr).Render() |

| Is Responsive | **Property:** IsResponsive <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").IsResponsive(true).Render();} | No separate Property for responsive, provided default. |

| Maximum Height | **Property:** MaxHeight <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").MaxHeight("500px").Render();} | Not Applicable |

| Maximum Width | **Property:** MaxWidth <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").MaxWidth("500px").Render();} | Not Applicable |

| Minimum Height | **Property:** MinHeight <br/>  
 <br/>@Html.EJ().RichTextEditor("rteSample").MinHeight("300px").Render();} | Not Applicable |

| Minimum Width | **Property:** MinWidth <br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").MinWidth("300px").Render();} | Not Applicable |

| name | **Property:** name <br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").name("commentBlog").Render();} | Not Applicable |

| Clear All | **Property:** ShowClearAll <br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowClearAll(true).Render();} | Not Applicable |

| Clear Format | **Property:** ShowClearFormat <br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowClearFormat(true).Render();} | No separate Property for clear Format. Its Provided in toolbar command.<br/> <br/> ViewBag.item = new[]  
 {"ClearFormat"} <br/> <br/>@Html.EJS().RichTextEditor("default").ToolBarSettings(e =>  
 e.Items((object)ViewBag.item)).Render() |

| Place holder | Not Applicable | **Property:** Placeholder<br/>  
 <br/>@Html.EJS().RichTextEditor("default").Placeholder("Enter the text").Render() |

| Context Menu | **Property:** ShowContextMenu<br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowContextMenu(true).Render();} | Not Applicable |

| Dimensions | **Property:** ShowDimensions<br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowDimensions(true).Render();} | Not Applicable |

| Show Footer | **Property:** ShowFooter<br/>  
 <br/><@{Html.EJ().RichTextEditor("rteSample").ShowFooter(true).Render();} | Not Applicable |

| Html Source | **Property:** ShowHtmlSource<br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowHtmlSource(true).Render();} | **Method:**  
 showSourceCode() <br/> <br/> var rteObj =  
 document.getElementById("richtexteditor").ej2\_Instances[0];<br/>  
 <br/>rteObj.showSourceCode(); |

| Html Tag Info | **Property:** ShowHtmlTagInfo<br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowHtmlTagInfo(true).Render();} | Not Applicable |

| Rounded Corner | **Property:** ShowRoundedCorner<br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").ShowRoundedCorner(true).Render();} | Not Applicable |

| Tooltip | **Property:** TooltipSettings<br/>  
 <br/>@{Html.EJ().RichTextEditor("rteSample").TooltipSettings(tooltip=>tooltip.ShowShadow(false)).Render();} | Not Applicable |

| value | **Property:** Value<br/> <br/>@{Html.EJ().RichTextEditor("rteSample").Value("The Rich Text Editor (RichTextEditor) control is an easy to render in client side.").Render();} | **Property:** Value<br/>

<br/>@Html.EJS().RichTextEditor("types").Value((string)ViewBag.value).Height("720px").ToolBarSettings(e => { e.Enable(true); }).Render() |

| Validation Rules | **Property:** ValidationRules<br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ValidationRules(new Dictionary<string, object> { { "required", "true" }, { "minWordCount", 15 } }).Render();} | Achieved in sample level. <br/>

<br/><https://ej2.syncfusion.com/aspnetmvc/documentation/rich-text-editor/validation.html#validation-rules> |

| Validation Message | **Property:** ValidationMessage<br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ValidationMessage(new Dictionary<string, object> { { "minWordCount", "A minimum of {10} words is required here" }, { "Required", "Please enter the content" } }).Render();} | Achieved in sample level.<br/>

<br/><https://ej2.syncfusion.com/aspnetmvc/documentation/rich-text-editor/validation.html#validation-message> |

| Zoom Step | **Property:** ZoomStep<br/>

<br/>@{Html.EJ().RichTextEditor("rteSample").ZoomStep("0.5").Render();} | Not Applicable |

| Disable | **Method:** disable() <br/> <br/> var rteObj = \$("#rte ").data("ejRTE "); <br/>

<br/>rteObj.disable() | **Property:** Enabled<br/>

<br/>@Html.EJS().RichTextEditor("default").Enabled(false).Render() |

| Enable | **Method:** enable()<br/> <br/> var rteObj = \$("#rte ").data("ejRTE ");<br/> <br/>

rteObj.enable() | **Property:** Enabled<br/>

<br/>@Html.EJS().RichTextEditor("default").Enabled(true).Render() |

| Focus | **Method:** focus() <br/> <br/> var rteObj = \$("#rte ").data("ejRTE ");<br/> <br/> rteObj.

focus() | **Method:** focusIn() <br/> <br/> var rteObj = document.getElementById("richtexteditor").ej2\_Instances[0]; <br/> <br/>rteObj. focusIn(); |

| Focus Out | Not Applicable | **Method:** focusOut() <br/> <br/>var rteObj = document.getElementById("richtexteditor").ej2\_Instances[0]; <br/> <br/>rteObj. focusOut(); |

| Command Status | **Method:** getCommandStatus() <br/> <br/>var rteObj = \$("#rte ").data("ejRTE ");<br/> <br/>rteObj.getCommandStatus() | Not Applicable |

| Get Document | **Method:** getDocument()<br/> <br/> var rteObj = \$("#rte ").data("ejRTE ");<br/> <br/> rteObj.getDocument() | Not Applicable |

| Get Html | **Method:** getHtml() <br/> <br/> var rteObj = \$("#rte ").data("ejRTE ");<br/> <br/> rteObj.getHtml() | **Method:** getHtml() <br/> <br/> var rteObj = document.getElementById("richtexteditor").ej2\_Instances[0]; <br/> <br/>rteObj. getHtml(); |

| Get Text | **Method:** getText() <br/> <br/> var rteObj = \$("#rte ").data("ejRTE "); <br/> <br/>rteObj.getText() | **Method:** getContent() <br/> <br/>var rteObj = document.getElementById("richtexteditor").ej2\_Instances[0]; <br/> <br/>rteObj. getContent(); |

| Get Selected Html | **Method:** getSelectedHtml()<br/> <br/> var rteObj = \$("#rte ").data("ejRTE ");<br/> <br/>rteObj.getSelectedHtml() | **Method:** getSelection() <br/> <br/>var rteObj =

```

document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.getSelection();
|
| Get Range | Not Applicable | Method: getRange()

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.getRange(); |
| Hide | Method: hide()

var rteObj = $("#rte ").data("ejRTE ");

rteObj.hide()
| Not Applicable |
| Show | Method: show()

var rteObj = $("#rte ").data("ejRTE ");

rteObj.show() | Not Applicable |
| Insert Menu Option | Method: insertMenuOption()

var rteObj = $("#rte ").data("ejRTE
");

rteObj.insertMenuOption({newItem:"Show Table Details",targetItem:"Table
Properties", insertType:("insertAfter"),
menuType:{text:false,image:false,hyperlink:false,table:true},spriteCssClass:"e-rte-toolbar-icon
tableProperties"}) | Not Applicable |
| Paste Content | Method: pasteContent(html)

var rteObj = $("#rte ").data("ejRTE
");

rteObj.pasteContent("Rich Text Editor content") | Not Applicable |
| refresh | Method: refresh()

var rteObj = $("#rte ").data("ejRTE ");

rteObj.refresh() | Method: refresh()

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.refresh(); |
| Select All | Method: selectAll()

 var rteObj = $("#rte ").data("ejRTE ");

rteObj.selectAll() | Method: selectAll()

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.selectAll(); |
| Select Range | Method: selectRange()

var rteObj = $("#rte ").data("ejRTE ");

rteObj.selectRange() | Method: selectRange()

 var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.selectRange(); |
| Set Html | Method: setHtml()

var rteObj = $("#rte ").data("ejRTE ");

rteObj.setHtml("richtexteditor content") | Not Applicable |
| Destroy | Not Applicable | Method: destroy()

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.destroy(); |
| Change | Event: Change

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt =>
evt.Change("onChange")).Render();} | Event: Change

@Html.EJS().RichTextEditor("defaultRTE").Change("onChange").Render() |
| create | Event: Create

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt
=> evt.Create("onCreate")).Render();} | Event: Created

@Html.EJS().RichTextEditor("defaultRTE").Created("onCreated").Render() |
| Context Menu Click | Event: ContextMenuClick

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt =>
evt.ContextMenuClick("onContextMenuClick")).Render();} | Not Applicable |

```

```
| destroy | Event: Destroy

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt =>
evt.Destroy("onDestroy")).Render();} | Event: Destroyed

@Html.EJS().RichTextEditor("defaultRTE").Destroyed("onDestroyed").Render() |

| Pre Render | Event: PreRender

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt =>
evt.PreRender("onPreRender")).Render();} | Not Applicable |

| Select | Event: Select

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt
=> evt.Select("onSelect")).Render();} | Not Applicable |

| Focus | Not Applicable | Event: Focus

@Html.EJS().RichTextEditor("defaultRTE").Focus("onFocus").Render() |
```

### Execute Command

```
| Behavior | API in Essential JS 1 | API in Essential JS 2 |
|-----|-----|-----|

| Execute Command | Method: executeCommand(cmdName, args, [textnodeType])

var
rteObj = $("#rte ").data("ejRTE ");

 rteObj.executeCommand("bold", true) | Method:
executeCommand(cmdName, value)

var rteObj =
document.getElementById("richtexteditor").ej2_Instances[0];

rteObj.
executeCommand("Bold"); |

| Execute events | Event: Execute

@{Html.EJ().RichTextEditor("rteSample").ClientSideEvents(evt =>
evt.Execute("onExecute")).Render();} | Event: ActionComplete

@Html.EJS().RichTextEditor("defaultRTE").ActionComplete("onActionComplete").Render()
|

| Before Execute | Not Applicable | Event: ActionBegin

@Html.EJS().RichTextEditor("defaultRTE").ActionBegin("onActionBegin").Render() |
```

## Schedule

### Getting Started with ASP.NET MVC Scheduler Control

This section briefly explains about how to include [ASP.NET MVC Scheduler](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)



### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

## Add ASP.NET MVC Scheduler control

Now, add the Syncfusion ASP.NET MVC Scheduler control in ~/Views/Home/Index.cshtml page.

**CSHTML**

```
@Html.EJS().Schedule("schedule").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Scheduler control will be rendered in the default web browser.

|          | May 01 - 07, 2022 ▾ |          |          |          |          |          |          | Today | Day | Week | Work Week | Month | Agenda |
|----------|---------------------|----------|----------|----------|----------|----------|----------|-------|-----|------|-----------|-------|--------|
|          | Sun<br>1            | Mon<br>2 | Tue<br>3 | Wed<br>4 | Thu<br>5 | Fri<br>6 | Sat<br>7 |       |     |      |           |       |        |
| 12:00 AM |                     |          |          |          |          |          |          |       |     |      |           |       |        |
| 1:00 AM  |                     |          |          |          |          |          |          |       |     |      |           |       |        |
| 2:00 AM  |                     |          |          |          |          |          |          |       |     |      |           |       |        |
| 3:00 AM  |                     |          |          |          |          |          |          |       |     |      |           |       |        |
| 4:00 AM  |                     |          |          |          |          |          |          |       |     |      |           |       |        |

## Populating appointments

To populate an empty Scheduler with appointments, bind the event data to it by assigning the [DataSource](#) property under [EventSettings](#) property.

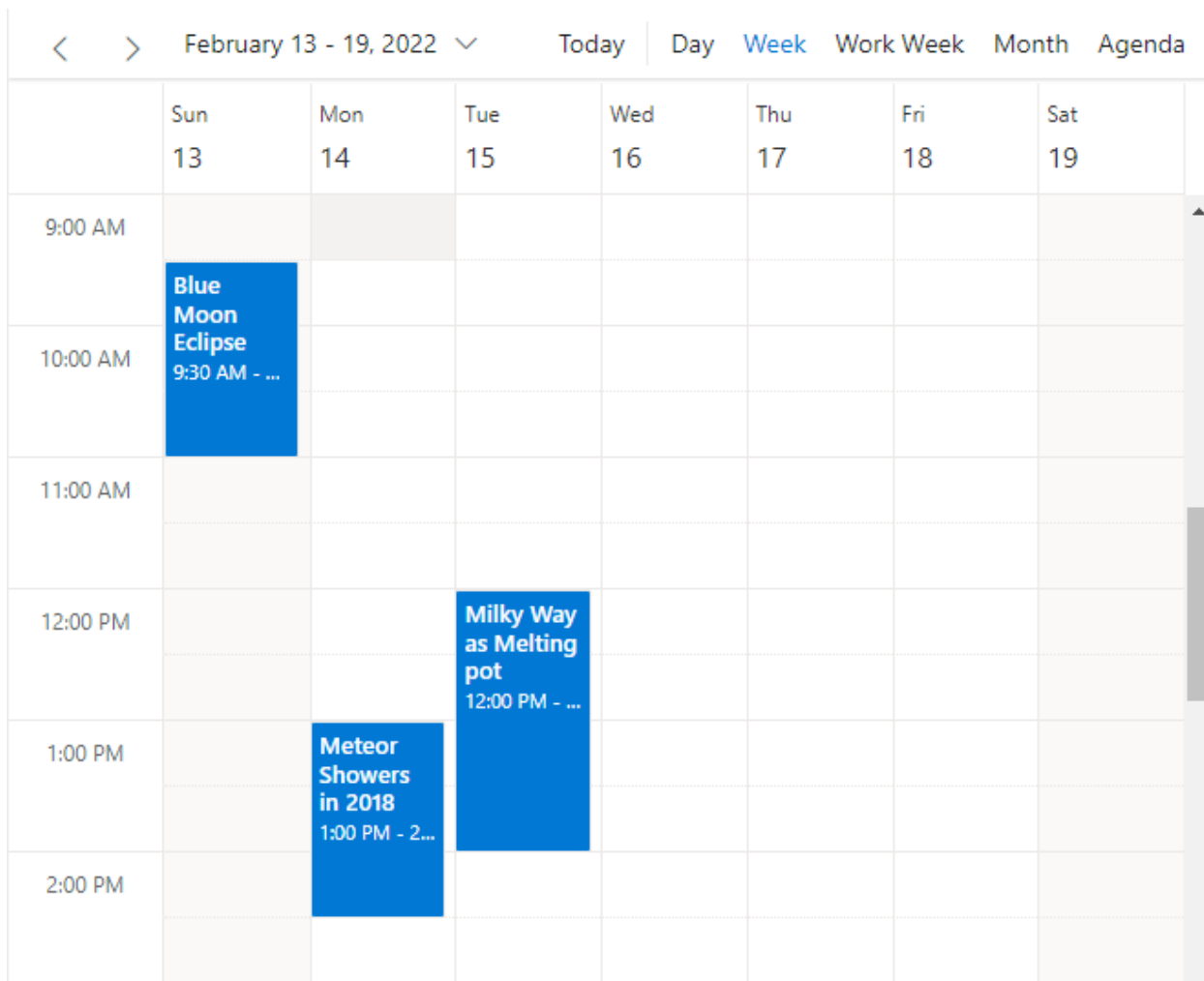
**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@model List<ScheduleSample.Controllers.AppointmentData>
```

```
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource = Model })
 .SelectedDate(new DateTime(2022, 2, 15))
 .Render()
)
```

### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(GetScheduleData());
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2022, 2, 11, 9, 30, 0), EndTime = new DateTime(2022, 2, 11, 11, 0,
 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", StartTime = new DateTime(2022,
 2, 12, 12, 0, 0), EndTime = new DateTime(2022, 2, 12, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2022, 2,
 13, 9, 30, 0), EndTime = new DateTime(2022, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2018", StartTime = new DateTime(2022,
 2, 14, 13, 0, 0), EndTime = new DateTime(2022, 2, 14, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", StartTime = new
 DateTime(2022, 2, 15, 12, 0, 0), EndTime = new DateTime(2022, 2, 15, 14, 0,
 0) });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```



### Setting date

Scheduler usually displays the system date as its current date. To change the current date of scheduler with specific date, define the [SelectedDate](#) property.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@model List<ScheduleSample.Controllers.AppointmentData>
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource = Model })
 .SelectedDate(new DateTime(2022, 2, 15))
 .Render()
)
```

### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(GetScheduleData());
 }
}
```

```

}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2022, 2, 11, 9, 30, 0), EndTime = new DateTime(2022, 2, 11, 11, 0,
 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", StartTime = new DateTime(2022,
 2, 12, 12, 0, 0), EndTime = new DateTime(2022, 2, 12, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2022, 2,
 13, 9, 30, 0), EndTime = new DateTime(2022, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2018", StartTime = new DateTime(2022,
 2, 14, 13, 0, 0), EndTime = new DateTime(2022, 2, 14, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", StartTime = new
 DateTime(2022, 2, 15, 12, 0, 0), EndTime = new DateTime(2022, 2, 15, 14, 0,
 0) });
 return appData;
}
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Setting view

Scheduler displays **week** view by default. To change the current view, define the applicable view name to the [CurrentView](#) property. The applicable view names are,

- Day
- Week
- WorkWeek
- Month
- Year
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@model List<ScheduleView>

```

```
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Views(Model)
 .Height("550px")
 .SelectedDate(new DateTime(2022, 2, 15))
 .Render()
)
```

### HOMECONTROLLER.CS

```
public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineDay }
 };
 return View(viewOption);
}
```

|          | February 13 - 19, 2022 ▾ |           |           |           |           |           |           | Today | Week | Timeline Day |
|----------|--------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-------|------|--------------|
|          | Sun<br>13                | Mon<br>14 | Tue<br>15 | Wed<br>16 | Thu<br>17 | Fri<br>18 | Sat<br>19 |       |      |              |
| 7:00 AM  |                          |           |           |           |           |           |           |       |      |              |
| 8:00 AM  |                          |           |           |           |           |           |           |       |      |              |
| 9:00 AM  |                          |           |           |           |           |           |           |       |      |              |
| 10:00 AM |                          |           |           |           |           |           |           |       |      |              |
| 11:00 AM |                          |           |           |           |           |           |           |       |      |              |

#### Individual view customization

Each individual scheduler views can be customized with its own options such as setting different start and end hour on Week and Work Week views, whereas hiding the weekend days on Month view alone. This can be achieved by defining views property to accept the array of object type, where each object depicts the individual view customization.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@model List<ScheduleView>
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(Model)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

**HOMECONTROLLER.CS**

```
public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week, DateFormat = "dd-MMM-yyyy" },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month, ShowWeekend = false, Readonly = true }
 };
 return View(viewOption);
}
```

| < > 13-Feb-2022 - 19-Feb-2022 ▾ |     |     |     |     |     |     |     | Today | Week | Month |
|---------------------------------|-----|-----|-----|-----|-----|-----|-----|-------|------|-------|
|                                 | Sun | Mon | Tue | Wed | Thu | Fri | Sat |       |      |       |
|                                 | 13  | 14  | 15  | 16  | 17  | 18  | 19  |       |      |       |
| 9:00 AM                         |     |     |     |     |     |     |     |       |      |       |
| 10:00 AM                        |     |     |     |     |     |     |     |       |      |       |
| 11:00 AM                        |     |     |     |     |     |     |     |       |      |       |
| 12:00 PM                        |     |     |     |     |     |     |     |       |      |       |

**Note:** [View Sample in GitHub.](#)

**Note:** You can also explore our [ASP.NET MVC Scheduler example](#) that shows how to use the toolbar buttons to play with Scheduler functionalities.

## ASP.NET MVC Scaffolding

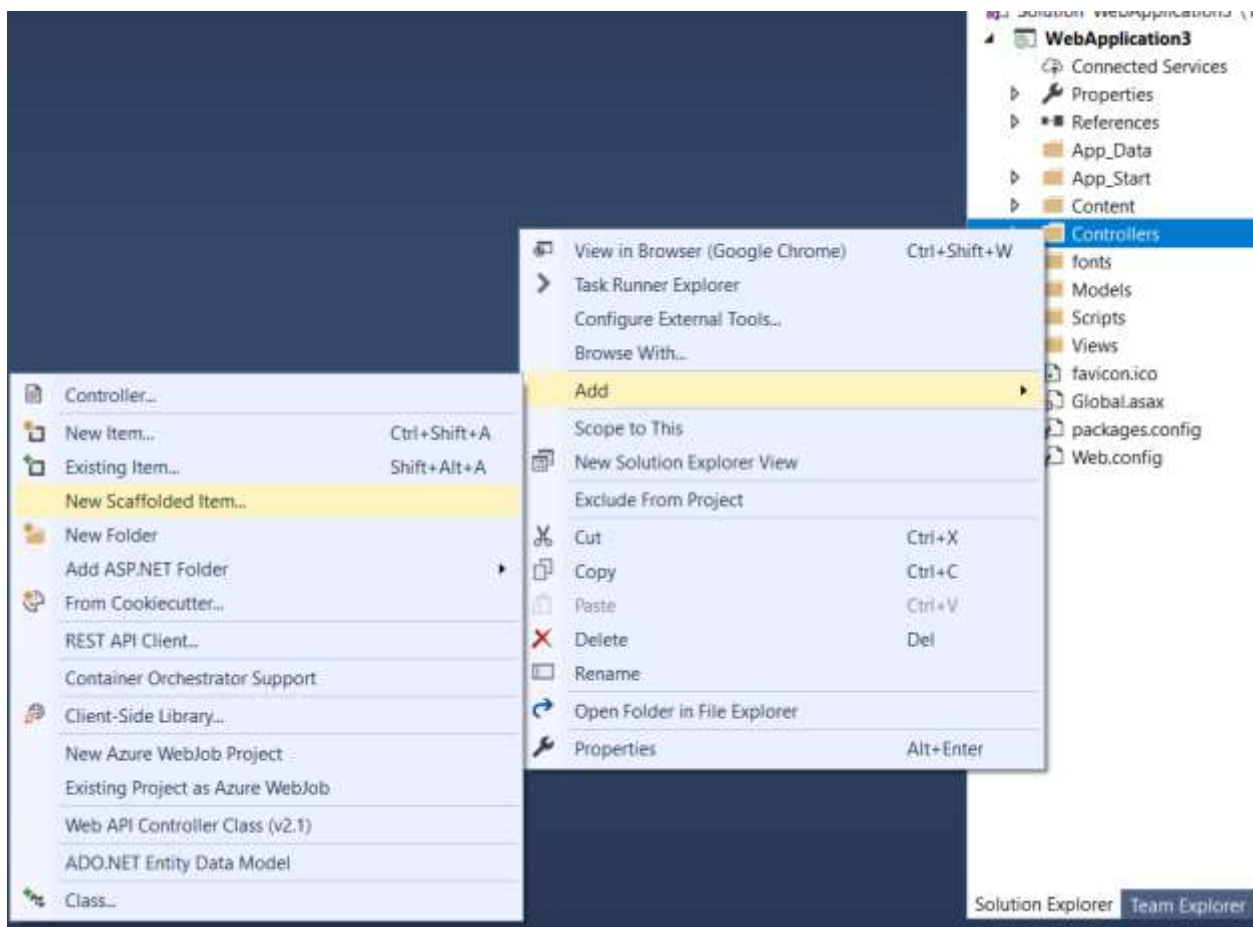
Syncfusion includes an extension for **Visual Studio** with UI Scaffolding options for the ASP.NET MVC Scheduler to quickly add its code and interact with data models. This allows you to easily create the appropriate **Views** and **Controller** action methods with respective ASP.NET MVC Scheduler code.

**Note:** The Syncfusion ASP.NET MVC UI Scaffolder is available from the version **v16.4.0.40**.

### Getting Started

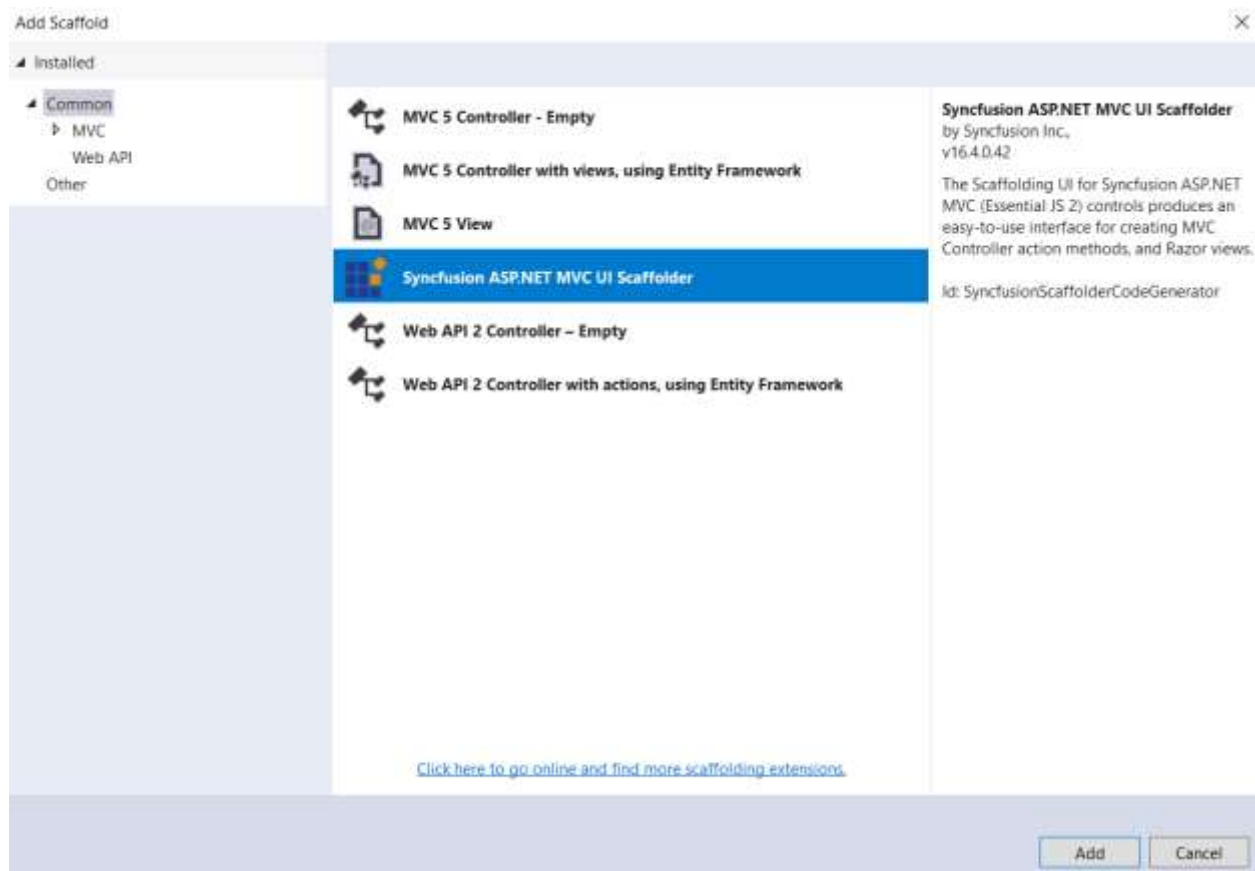
Let's start with the steps on how to scaffold the ASP.NET MVC Scheduler into your web application.

- Create an ASP.NET MVC application and add an Entity Framework data model referring from the [documentation](#) with Scheduler related fields such as Id, Subject, Location, Start Date, End Date and All-day. Once the model file is added, ensure the required DbContext and all its related properties are added.
- Refer the [Getting Started documentation](#) to know about how to configure the Syncfusion Essential JS2 for ASP.NET MVC in your web application.
- Right-click on the **Controllers** folder in the Solution Explorer and select **Add → New Scaffolded Item** from the menu options.

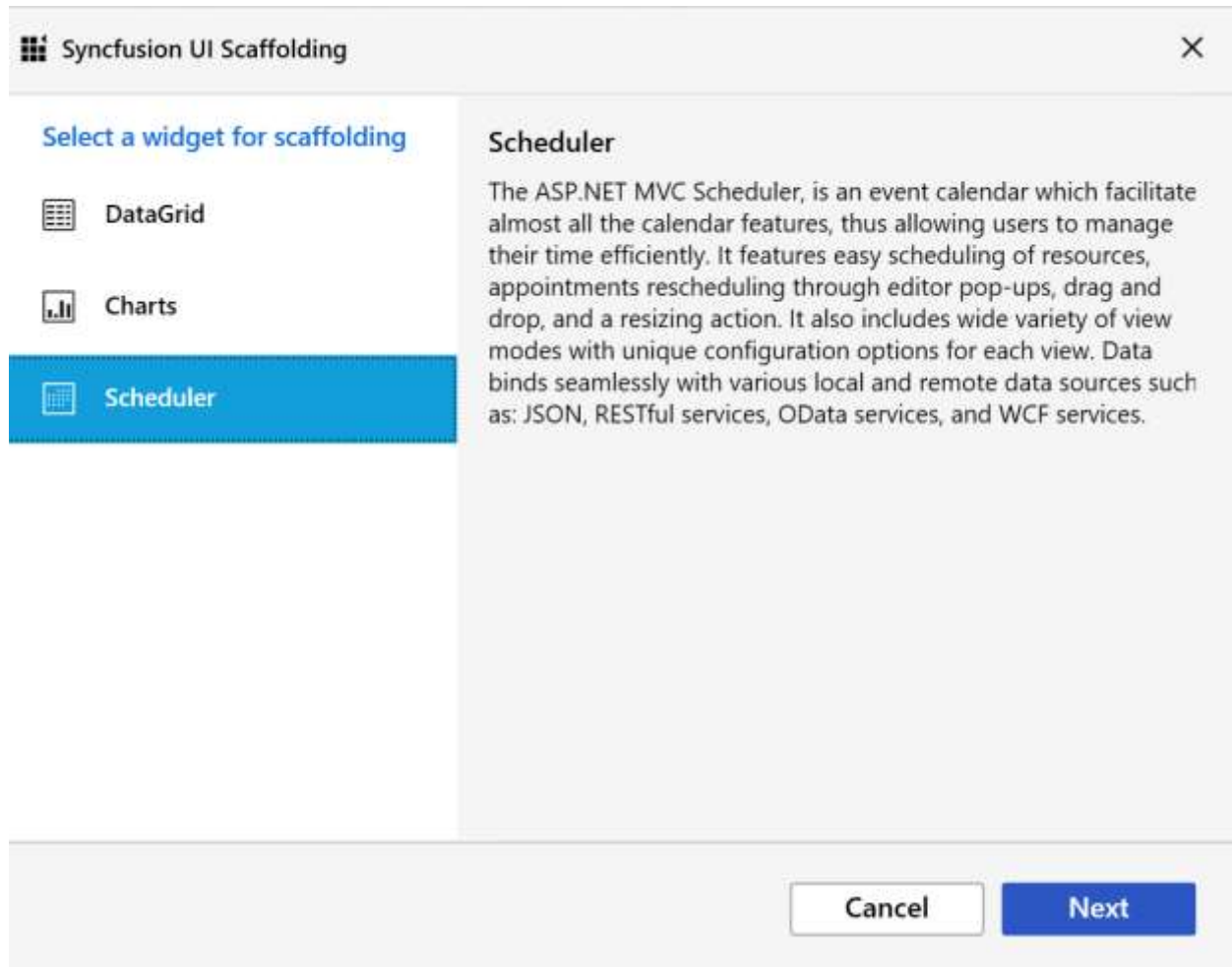


- You will see the **Add Scaffold** dialog. Select the **Syncfusion ASP.NET MVC UI Scaffolder** and click **Add** button, which will display the Syncfusion UI Scaffolding dialog.

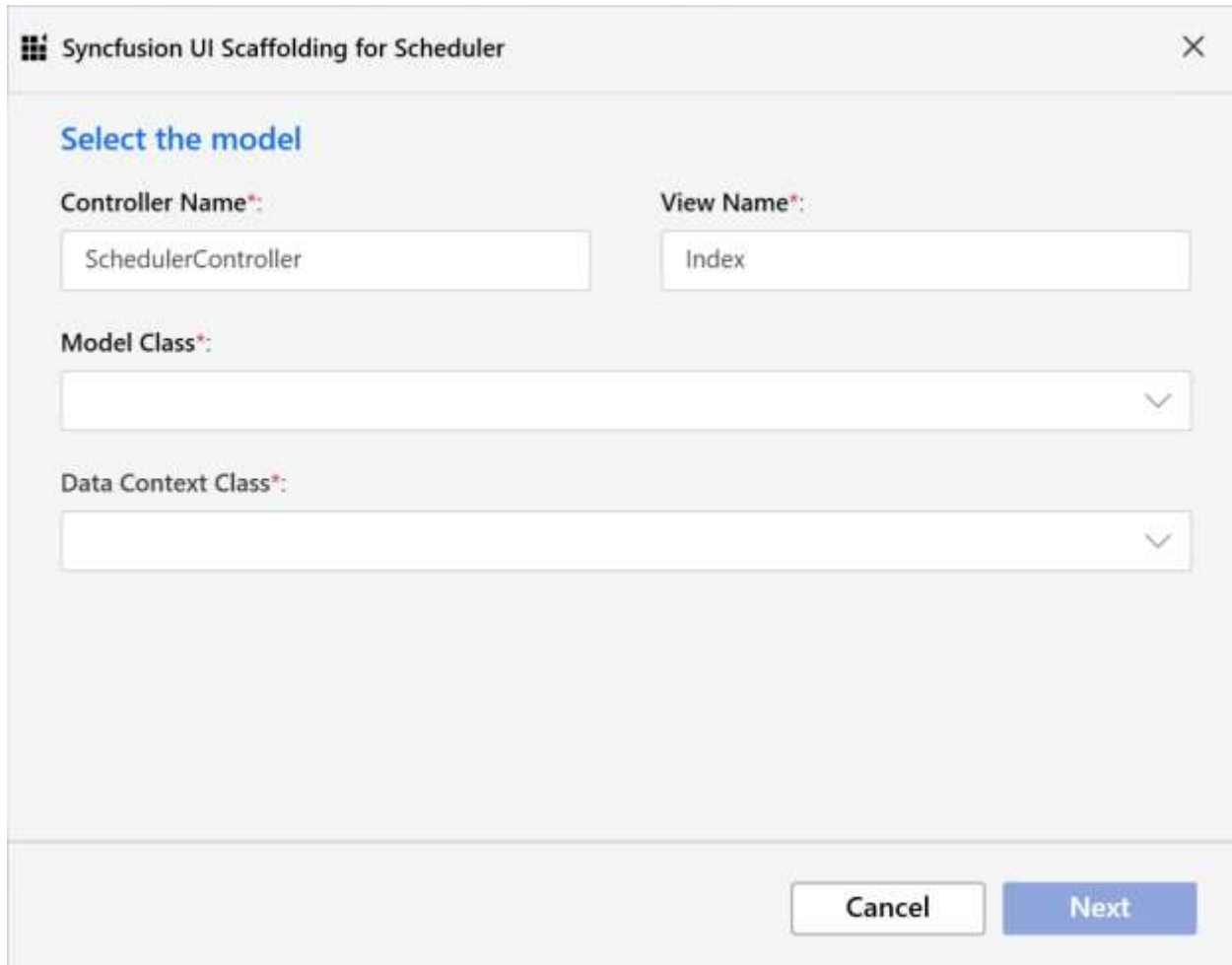




- Choose the Scheduler control to perform Scaffolding with it and click **Next**.



- Syncfusion UI Scaffolding for Scheduler dialog will be opened, from which you are opted to choose the Model and Data Context options. Enter the **Controller** and **View** names as per the application requirement. Once the required **Model Class** and its relevant **Data Context Class** are chosen, now click the **Next** button, which offers the Scheduler functionalities to be configured before scaffolding.



**Syncfusion UI Scaffolding for Scheduler**

**Select the model**

**Controller Name\*:** SchedulerController

**View Name\*:** Index

**Model Class\*:**

**Data Context Class\*:**

Cancel Next

**Note:** All the model types present in the current application will be listed in the **Model Class** DropDownList. Also, from the available **Data Context Class**, choose the appropriate Entity Framework Data Model.

- Now, select the required Scheduler options (select the corresponding **Scheduler Views** and **Properties** from the options) and Click the **Add** button. Use the **Back Arrow**, if you need to modify the already chosen Controller or View name, or to change the **selected Model Class** and **Data Context Class**.

**Syncfusion UI Scaffolding for Scheduler**

**Select the feature(s)**

DataSource Type:

Local Data

☐ TimeScale ☐ ReadOnly

**Scheduler Views\*:**

☐ Day ☐ Week ☐ WorkWeek

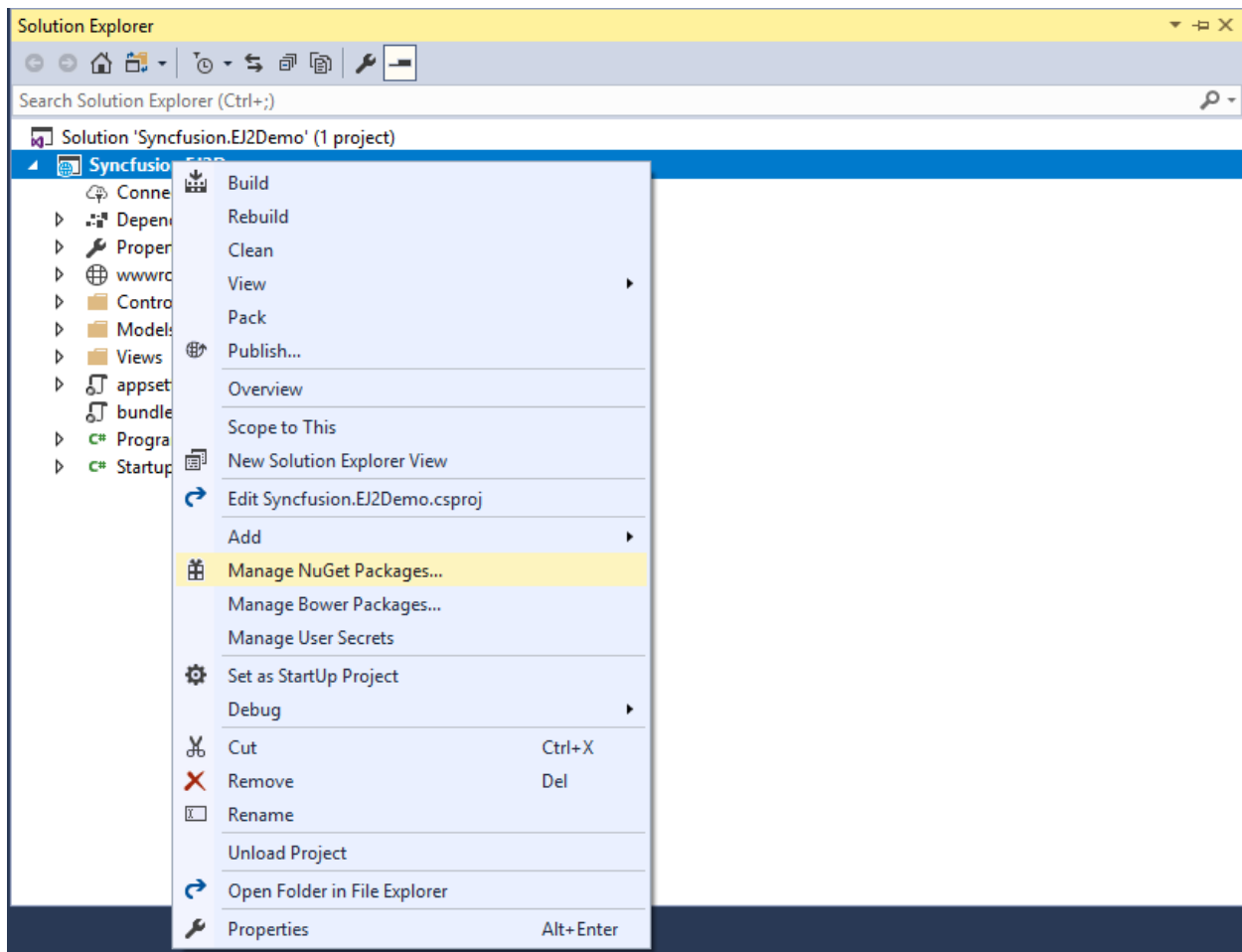
☐ Month ☐ Agenda ☐ MonthAgenda

☐ TimelineDay ☐ TimelineWeek ☐ TimelineWorkWeek

☐ TimelineMonth

← Cancel Add

- Once the required Scheduler options are configured through the **Scheduler UI Scaffolding**, the respective Scheduler **Controller** and the corresponding **View** files are now generated with the appropriate Scheduler code snippet.



**Note:** Ensure that at least one Entity Framework model exists in your active project and also the application gets compiled once. If you make any changes in the Model properties later, compile the application once before performing scaffold.

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Scheduler interactions

The following table describes the Scheduler actions and also illustrates how those actions are carried out through mouse and touch interactions on Scheduler.

| Actions                      | Mouse interaction                                                                               | Touch interaction                                                                                                      |
|------------------------------|-------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| ----- -----  -----           |                                                                                                 |                                                                                                                        |
| Single click or tap on cells | Single click on a cell to select a cell.                                                        | Single tapping on cells, will display a + icon on the cell. Tapping on it again will open the new event editor window. |
| Multiple cell selection      | Single click on a cell and drag the selection to other cells to enable multiple cell selection. | No multiple cell selection is allowed using touch gestures.                                                            |

| Event selection | Single click on an event to select it. | Tap holding on events, select an event and opens a small popup at the top holding the options to edit or delete. The popup also displays the selected event's subject. |

| Multiple event selection and deletion | Pressing **Ctrl** key and altogether single clicking on multiple events one after the other will enable multiple event selection. Pressing **Delete** key after event selection will delete all the selected events. | Tap hold an event to select it, which opens a small popup at the top holding the options to edit or delete. As a continuation of this action, keep on single tapping on other events, to enable multiple event selection. Also, the popup displayed at the top remains in opened state, showing the count of the number of selected events. Pressing **Delete** option from the popup will delete all the selected events. |

| Date navigation | Clicking on the previous or next date navigation icons in the header bar allows you to navigate between dates. | Swiping the scheduler view port to the left or right will allow you to navigate between the dates on touch devices. You can prevent the swiping action by disabling [allowSwiping](#) property. NOTE: Swiping does not work when horizontal scroller present in the Scheduler. You can also make use of the previous and next navigation icons at the header bar to navigate. |

| View navigation | Click on an event and try moving it over the Scheduler to enable drag and drop action. | The view options are available within the popup options at the top right extreme end of the header bar and you can choose the view from it. |

| Drag and drop | Click on an event and try moving it over the Scheduler to enable drag and drop action. | Tap hold the event and try moving it over the Scheduler to enable drag and drop action. |

| Event resizing | Hover the mouse across the extremities or edges of the Scheduler events and when the mouse pointer changes into resize handler, now click and start resizing an event to the desired time range. | Touch the event extremities and start resizing the events directly. |

| Tooltip | Hover the mouse pointer over the events or resource header and the tooltip will be displayed. | Tap holding the events will open the tooltip on events. |

| Open editor window | Double click on cells or events to open the editor window. | Double click on cells or events to open the editor window. Single tap on cells, which displays a **+** icon on the cell. Now, tap on it again to open the new event editor window. To open the editor on events, single tap on it and then click on the edit icon to open the editor window in **Edit** mode. |

| Open quick info popup | Single clicking on a cell will open a quick popup prompting for new event creation. Single clicking on an event will open a popup displaying event information along with the option to edit and delete it. | No quick info popup is available while single tapping on cells. Single tapping on events, opens the popup showing event information. |

### Appointments in ASP.NET MVC Schedule Component

Appointments can be anything that are scheduled for a specific time period. It can be created on varied time range and each appointments are categorized based on this range. The Scheduler events can be categorized as,

- Normal events
- Spanned events
- All-day events
- Recurring events

### Normal events

Represents an appointment that is created for any specific time interval within a day.

#### Creating a normal event

The following example depicts how to define a normal event on the Scheduler, with event data being loaded from simple JSON data.

#### CHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Subject = "Paris",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Spanned events

Represents an appointment that is created for more than 24 hours, and usually displayed on the all-day row. Also, represents another type of appointment that is created for more than one day but less than 24 hours, and usually displayed appropriately on both the days.

**Note:** For example, if an appointment is created for two days say from November 25, 2018 – 11.00 PM to November 26, 2018 2.00 AM but less than 24 hours time interval, then the appointment is split into two partitions and will be displayed on both the days.

### All-day events

Represents an appointment that is created for an entire day such as holiday events. It is usually displayed separately in an all-day row, a separate row for all-day appointments below the date header section. In Timeline views, the all-day appointments displays in the working space area, and no separate all-day row is present in that view.

**Note:** To change normal appointment into all-day event, set `IsAllDay` field to true.

### Hide all-day row events

You can make use of the CSS customization to prevent the display of all-day row appointments on the Scheduler UI.

```
`css
.e-schedule .e-date-header-wrap .e-schedule-table thead {
display: none;
}
`
```

**Note:** You can also enable scroller for all-day row, [refer](#) here to know more.

### Customize the rendering of the spanned events

By default, Scheduler will renders the spanned events (appointment with more than 24 hours duration) in the all-day row by setting `AllDayRow` will the default type renders to the `SpannedEventPlacement` option within the `EventSettings` property. Now we can customize rendering of the that events inside the work cells itself by modifying the `SpannedEventPlacement` option as `TimeSlot`. In this following example, shows how to render the spanned appointments inside the work cells as follows.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource, SpannedEventPlacement = SpannedEventPlacement.TimeSlot
}))
 .SelectedDate(new DateTime(2018, 1, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
```



```

 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 17, 12, 30, 0),
 IsAllDay = false
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "London",
 StartTime = new DateTime(2018, 1, 16, 12, 0, 0),
 EndTime = new DateTime(2018, 1, 18, 13, 0, 0),
 IsAllDay = false
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

### Recurring events

Represents an appointment that is created for a certain time interval and occurring repeatedly on a daily, weekly, monthly or yearly basis at the same time interval based on the provided recurrence rule. Usually, the recurring events are indicated by a repeat marker added at the bottom-right position.

### Creating a recurring event

The following example depicts how to create a recurring event on Scheduler with the specific recurrence rule. In the following example, an event is made to repeat on daily mode and ends after 5 occurrences.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}

public List<AppointmentData> GetScheduleData()
{

```

```

List<AppointmentData> appData = new List<AppointmentData>();
appData.Add(new AppointmentData
{
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 RecurrenceRule = "FREQ=DAILY; INTERVAL=1; COUNT=5"
});
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string RecurrenceRule { get; set; }
}

```

### Adding exceptions

A few instance of the recurrence series can be excluded on specific dates, by adding those exceptional dates to the **RecurrenceException** field. These date values should be given in the ISO date time format with no hyphens(-) separating the date elements.

For example, 22nd February 2018 can be represented as 20180222. Also, the time part being represented in UTC format needs to add "Z" after the time portion with no space. "07:30:00 UTC" is therefore represented as "073000Z".

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),

```

```

 EndTime = new DateTime(2018, 1, 28, 12, 30, 0),
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=8",
 RecurrenceException =
"20180129T043000Z,20180131T043000Z,20180202T043000Z"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string RecurrenceRule { get; set; }
 public string RecurrenceException { get; set; }
}

```

#### *Editing an occurrence from a series*

To dynamically edit a particular occurrence from an event series and display it on the initial load of Scheduler, the edited occurrence needs to be added as a new event to the dataSource collection, with an additional **RecurrenceID** field defined to it. The **RecurrenceID** field of edited occurrence usually maps the ID value of the parent event.

In this example, a recurring instance that displays on the date 30th Jan 2018 is edited with different timings. Therefore, this particular date is excluded from the parent recurring event that repeats from 28th January 2018 to 4th February 2018. This can be done by adding the **RecurrenceException** field with the excluded date value on the parent event. Also, the edited occurrence event which is created as a new event should carry the **RecurrenceID** field pointing to the parent event's **Id** value.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Scrum Meeting",

```

```

 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0),
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=8",
 RecurrenceException = "20180130T043000Z"
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Scrum Meeting",
 StartTime = new DateTime(2018, 1, 30, 09, 0, 0),
 EndTime = new DateTime(2018, 1, 30, 10, 30, 0),
 RecurrenceID = 1
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string RecurrenceRule { get; set; }
 public string RecurrenceException { get; set; }
 public Nullable<int> RecurrenceID { get; set; }
}

```

#### *Edit only the current and following events*

To edit only the current and following events enable the property `editFollowingEvents` within `eventSettings` property. The edited occurrence needs to be added as a new event to the `dataSource` collection, with an additional `followingID` field defined to it. The `followingID` field of edited occurrence usually maps the ID value of the immediate parent event.

In this example, a recurring instance that displays on the date 30th Jan 2018 and its following dates are edited with different subject. Therefore, this particular date and its following dates are excluded from the parent recurring event that repeats from 28th January 2018 to 4th February 2018. This can be done by updating the `recurrenceRule` field with the until date value on the parent event. Also, the edited events which is created as a new event should carry the `followingID` field pointing to the immediate parent event's `Id` value.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments, EditFollowingEvents = true })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()
{

```

```

 ViewBag.appointments = GetScheduleData();
 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Scrum Meeting",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0),
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;UNTIL=20180129T043000Z;"
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Scrum Meeting - Following Edited",
 StartTime = new DateTime(2018, 1, 30, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 30, 10, 30, 0),
 RecurrenceRule =
"FREQ=DAILY;INTERVAL=1;UNTIL=20180204T043000Z;",
 FollowingID = 1
 });
 return appData;
 }
 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string RecurrenceRule { get; set; }
 public string RecurrenceException { get; set; }
 public int FollowingID { get; set; }
 }

```

#### *Recurrence options and rules*

Events can be repeated on a daily, weekly, monthly or yearly basis based on the recurrence rule which accepts the string value. The following details should be assigned to the **RecurrenceRule** property to generate the recurring instances.

- Repeat type - daily/weekly/monthly/yearly.
- How many times it needs to be repeated?
- The interval duration.
- The time period to render the appointment, etc.

There are four repeat types available namely,

- **Daily** - Creates the recurring instances on daily basis.
- **Weekly** - Creates the recurring instances on weekly basis for the selected days.
- **Monthly** - Creates the recurring instances on monthly basis for the selected months and other provided recurrence criteria.

- **Yearly** - Creates the recurring instances on yearly basis.

#### *Recurrence properties*

The properties based on which the recurrence appointments are created with its respective time period are depicted in the following table. Also, the valid rule string can be referred from [iCalendar](#) specifications.

**Note:** Refer [iCalendar](#) specifications for valid recurrence rule string.

| Property   | Purpose                                                                                                                                                                                                                                                                                                                                                                                                | Example                                                 |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| FREQ       | Maintains the repeat type (Daily, Weekly, Monthly, Yearly) value of the appointment.                                                                                                                                                                                                                                                                                                                   | FREQ=DAILY;INTERVAL=1                                   |
| INTERVAL   | Maintains the interval value of the appointments. When you create the daily appointment at an interval of 2, the appointments are rendered on the days Monday, Wednesday and Friday (Creates an appointment on all days by leaving the interval of one day gap).                                                                                                                                       | FREQ=DAILY;INTERVAL=2                                   |
| COUNT      | It holds the appointment's count value. When the COUNT value is 10, then 10 instances of appointments are created in the recurrence series.                                                                                                                                                                                                                                                            | FREQ=DAILY;INTERVAL=1;COUNT=10                          |
| UNTIL      | This property holds the end date value (in ISO format) denoting when the recurrence actually ends.                                                                                                                                                                                                                                                                                                     | FREQ=DAILY;INTERVAL=1;UNTIL=20180530T041343Z            |
| BYDAY      | It holds the day value(s), representing on which the appointments actually renders. Create the weekly appointment, and select the day(s) from the day options (Monday/Tuesday/Wednesday/Thursday/Friday/Saturday/Sunday). When Monday is selected, the first two letters of the selected day "MO" is saved in the BYDAY property. When multiple days are selected, the values are separated by commas. | FREQ=WEEKLY;INTERVAL=1;BYDAY=MO,WE;COUNT=10             |
| BYMONTHDAY | This property is used to store the date value of the Month, while creating the Month recurrence appointment. When you create a Monthly recurrence appointment for every 3rd day of the month, then BYMONTHDAY holds the value 3 and creates the appointment on 3rd day of every month.                                                                                                                 | FREQ=MONTHLY;BYMONTHDAY=3;INTERVAL=1;COUNT=10           |
| BYMONTH    | This property is used to store the index value of the selected Month while creating the yearly appointments. When you create the yearly appointment on June month, the index value of June month 6 will get stored in the BYMONTH field. The appointment is created on every 6th month of a year.                                                                                                      | FREQ=YEARLY;BYMONTHDAY=16;BYMONTH=6;INTERVAL=1;COUNT=10 |
| BYSETPOS   | This property is used to store the index value of the week. When you create the monthly appointment in second week of a month, the index value of the second week (2) is stored in BYSETPOS.                                                                                                                                                                                                           | FREQ=MONTHLY;BYDAY=MO;BYSETPOS=2;COUNT=10               |

**Note:** The default recurrence related validation has been included for recurrence appointments similar to the one available in Outlook. The validation usually occurs during the recurrence appointment creation, editing, drag and drop or resizing of the recurrence appointments and also if any single occurrence changes.

#### *Daily Frequency*

| Description | Example |
|-------------|---------|
|             |         |

| Daily recurring event that never ends | FREQ=DAILY; INTERVAL=1 |

| Daily recurring event that ends after 5 occurrences | FREQ=DAILY; INTERVAL=1; COUNT=5 |

| Daily recurring event that ends exactly on 12/12/2018 | FREQ=DAILY; INTERVAL=1; UNTIL=12/12/2018 |

| Daily event that recurs on alternative days and repeats for 10 occurrences | FREQ=DAILY; INTERVAL=2; COUNT=10 |

#### *Weekly Frequency*

| Description | Example |

|-----|-----|

| Weekly recurring event that repeats on every Monday, Wednesday and Friday and never ends | FREQ=WEEKLY; INTERVAL=1; BYDAY=MO,WE,FR |

| Repeats every week Thursday and ends after 10 occurrences | FREQ=WEEKLY; INTERVAL=1; BYDAY=TH; COUNT=10 |

| Repeats every week Monday and ends on 12/12/2018 | FREQ=WEEKLY; INTERVAL=1; BYDAY=MO; UNTIL=12/12/2018 |

| Repeats on Monday, Wednesday and Friday of alternative weeks and ends after 10 occurrences | FREQ=WEEKLY; INTERVAL=2; BYDAY=MO, WE, FR; COUNT=10 |

#### *Monthly Frequency*

| Description | Example |

|-----|-----|

| Monthly recurring event that repeats on every 15th day of a month and never ends | FREQ=MONTHLY; BYMONTHDAY=15; INTERVAL=1 |

| Monthly recurring event that repeats on every 16th day of a month and ends after 10 occurrences | FREQ=MONTHLY; BYMONTHDAY=16; INTERVAL=1; COUNT=10 |

| Repeats every 17th day of a month and ends on 12/12/2018 | FREQ=MONTHLY; BYMONTHDAY=17; INTERVAL=1; UNTIL=12/12/2018 |

| Repeats every 2nd Friday of a month and never ends | FREQ=MONTHLY; BYDAY=FR; BYSETPOS=2; INTERVAL=1 |

| Repeats every 4th Wednesday of a month and ends after 10 occurrences | FREQ=MONTHLY; BYDAY=WE; BYSETPOS=4; INTERVAL=1; COUNT=10 |

| Repeats every 4th Friday of a month and ends on 12/12/2018 | FREQ=MONTHLY; BYDAY=FR; BYSETPOS=4; INTERVAL=1; UNTIL=12/12/2018; |

#### *Yearly Frequency*

| Description | Example |

|-----|-----|

| Yearly event that repeats on every 15th day of December month and never ends | FREQ=YEARLY; BYMONTHDAY=15; BYMONTH=12; INTERVAL=1 |

| Event that repeats on every 10th day of December month and ends after 10 occurrences |  
FREQ=YEARLY; BYMONTHDAY=10; BYMONTH=12; INTERVAL=1; COUNT=10 |

| Repeats on every 12th day of December month and ends on 12/12/2025 | FREQ=YEARLY;  
BYMONTHDAY=12; BYMONTH=12; INTERVAL=1; UNTIL=12/12/2025 |

| Repeats on every 3rd Friday of December month and never ends | FREQ=YEARLY; BYDAY=FR;  
BYMONTH=12; BYSETPOS=3; INTERVAL=1 |

| Repeats on every 3rd Tuesday of December month and ends after 10 occurrences | FREQ=YEARLY;  
BYDAY=TU; BYMONTH=12; BYSETPOS=3; INTERVAL=1; COUNT=10 |

| Repeats on every 4th Wednesday of December month and ends on 12/12/2028 | FREQ=YEARLY;  
BYDAY=WE; BYMONTH=12; BYSETPOS=4; INTERVAL=1; UNTIL=12/12/2028 |

### Recurrence Validation

The built-in validation support has been added by default for recurring appointments during its creation, edit, drag and drop or resize action. The following are the possible validation alerts that displays on Scheduler while creating or editing the recurring events.

| Validation messages | Description |

|-----|-----|

| The recurrence pattern is not valid. | This alert will raise, when the selected recurrence rule value is not a valid one. For example, when you try to select the end date value (using **Until** option) for a recurring event, which occurs before the start date, an alert will popup out saying that the chosen pattern is invalid. |

| The changes made to specific instances of this series will be cancelled and those events will match the series again. | This alert will raise, when you try to edit the whole series, whose occurrence might have been already edited. For example, If there are five occurrences and one of the occurrence is already edited. Now, when you try to edit the entire series, you will get this validation alert. |

| The duration of the event must be shorter than how frequently it occurs. Shorten the duration, or change the recurrence pattern in the recurrence event editor. | This validation will occur, if the event duration is longer than the selected frequency. For example, if you create a recurring appointment with two days duration in **Daily** frequency with no intervals set to it, you may get this alert. |

| Some months have fewer than the selected date. For these months, the occurrence will fall on the last date of the month. | When you try to create a recurring appointment on 31st of every month, where few months won't have 31 days and in this scenario, you will get this alert. |

| Two occurrences of the same event cannot occur on the same day. | This validation will occur, when you try to edit or move any single occurrence to some other date, where another occurrence of the same event is already present. |

### Event fields

The Scheduler dataSource usually holds the event instances, where each of the instance includes a collection of appropriate [fields](#). It is mandatory to map these fields with the equivalent fields of database, when remote data is bound to it. When the local JSON data is bound, then the field names defined within the instances needs to be mapped with the scheduler event fields correctly.

**Note:** To create an event on Scheduler, it is enough to define the **startTime** and **endTime**. Also **id** field becomes mandatory to process CRUD actions on appropriate events.



*Built-in fields*

The built-in fields available on Scheduler event object are as follows.

| Field name          | Description                                                                                                                                                                                                                                                                                                                                                             |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Id                  | The Id field needs to be defined as mandatory and this field usually assigns a unique ID value to each of the events.                                                                                                                                                                                                                                                   |
| Subject             | The Subject field is optional, and usually assigns the summary text to each of the events.                                                                                                                                                                                                                                                                              |
| StartTime           | The StartTime field defines the start time of an event and it is mandatory to provide it for any of the valid event objects.                                                                                                                                                                                                                                            |
| EndTime             | The EndTime field defines the end time of an event and it is mandatory to provide the end time for any of the valid event objects.                                                                                                                                                                                                                                      |
| StartTImezone       | It maps the StartTImezone field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the StartTime field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler. |
| EndTImezone         | It maps the EndTImezone field from the dataSource and usually accepts the valid IANA timezone names. It is assumed that the value provided for this field is taken into consideration while processing the EndTime field. When this field is not mapped with any timezone names, then the events will be processed based on the timezone assigned to the Scheduler.     |
| Location            | It maps the Location field from the dataSource and the location text value will be displayed over the events.                                                                                                                                                                                                                                                           |
| Description         | It maps the Description field from the dataSource and denotes the event description which is optional.                                                                                                                                                                                                                                                                  |
| IsAllDay            | The IsAllDay field is mapped from the dataSource and is used to denote whether an event is created for an entire day or for specific time alone. Usually, an event with IsAllDay field set to true will be considered as an all-day event.                                                                                                                              |
| RecurrenceID        | It maps the RecurrenceID field from dataSource and usually holds the ID value of the parent recurrence event. This field is applicable only for the edited occurrence events.                                                                                                                                                                                           |
| RecurrenceRule      | It maps the RecurrenceRule field from dataSource and holds the recurrence rule value in a string format. Also, it uniquely identifies whether the event belongs to a recurring type or normal ones.                                                                                                                                                                     |
| RecurrenceException | It maps the RecurrenceException field from dataSource and is used to hold the collection of exception dates, on which the recurring occurrences needs to be excluded. The RecurrenceException should be specified in UTC format.                                                                                                                                        |
| IsReadOnly          | It maps the IsReadOnly field from dataSource. It is mainly used to make specific appointments as readonly when set to true.                                                                                                                                                                                                                                             |
| IsBlock             | It maps the IsBlock field from dataSource. It is used to block the particular time ranges in the Scheduler and prevents the event creation on those time slots.                                                                                                                                                                                                         |

*Binding different field names*

When the fields of event instances has the default mapping name, it is not mandatory to map them manually. If a Scheduler's dataSource holds the events collection with different field names, then it is necessary to map them with its equivalent field name within the `EventSettings` property.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule").Height("550px")
 .EventSettings(es =>
 es.Fields(f =>
 f.Subject(sub => sub.Name("TravelSummary"))
 .Id("TravelId")
 .IsAllDay(allday => allday.Name("FullDay"))
 .Location(loc => loc.Name("Source"))
 .Description(des => des.Name("Comments"))
 .StartTime(st => st.Name("DepartureTime"))
 .EndTime(et => et.Name("ArrivalTime"))
 .StartTImezone(stz => stz.Name("Origin"))
 .EndTImezone(etz => etz.Name("Destination"))
)
).DataSource(ViewBag.appointments)
)
.SelectedDate(new DateTime(2018, 2, 15))
.Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 TravelId = 1,
 TravelSummary = "Paris",
 DepartureTime = new DateTime(2018, 2, 15, 10, 0, 0),
 ArrivalTime = new DateTime(2018, 2, 15, 12, 30, 0),
 FullDay = false,
 Source = "London",
 Comments = "Summer vacation planned for outstation.",
 Origin = "Asia/Yekaterinburg",
 Destination = "Asia/Yekaterinburg"
 });
 return appData;
}
public class AppointmentData
{
 public int TravelId { get; set; }
 public string TravelSummary { get; set; }
 public DateTime DepartureTime { get; set; }
```

```

public DateTime ArrivalTime { get; set; }
public bool FullDay { get; set; }
public string Source { get; set; }
public string Comments { get; set; }
public string Origin { get; set; }
public string Destination { get; set; }
}

```

**Note:** The mapper field **Id** is of string type and has no additional validation options, whereas all other fields are of **Object** type and has additional options.

#### Event field settings

Each field of the Scheduler events are provided with additional settings such as options to set default value, to map with appropriate data source fields, to validate every event fields and to provide label values for those fields in the event window.

| Options    | Description                                                                                                                                  |
|------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Default    | Accepts the default value to the applicable fields (Subject, Location and Description), when no values are provided to them from dataSource. |
| Name       | Accepts the field name to be mapped from the dataSource fields.                                                                              |
| Title      | Accepts the label values to be displayed for the fields of event editor.                                                                     |
| Validation | Defines the validation rules to be applied on the event fields within the event editor.                                                      |

In following example, the Subject field in event editor will display its appropriate label as **Summary**. When no subject value is provided while saving an event, then the appointment will be saved with the default subject value as **Add Summary**.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(e => e.Fields(f => f.Id("Id")
 .Subject(sub =>
 sub.Name("Subject").Title("Summary").Default("Add Summary")
 .Location(loc => loc.Name("Location"))
 .Description(des => des.Name("Description"))
 .StartTime(st => st.Name("StartTime"))
 .EndTime(et => et.Name("EndTime"))
)
 .DataSource(ViewBag.appointments)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
)

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
}

```

```

 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Subject = "Paris",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
 });
 return appData;
 }
 public class AppointmentData
 {
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

### Adding Custom fields

Apart from the default Scheduler fields, the user can include 'n' number of custom fields for appointments. The following code example shows how to include two custom fields namely **Status** and **Priority** within event collection. It is not necessary to bind the custom fields within the **EventSettings**. However, those additional fields can be accessed easily, for internal processing as well as from application end.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(e => e.Fields(f => f.Subject(sub => sub.Name("Subject"))
 .Id("Id")
 .IsAllDay(allday => allday.Name("IsAllDay"))
 .StartTime(st => st.Name("StartTime"))
 .EndTime(et => et.Name("EndTime"))
)
 .DataSource(ViewBag.appointments)
)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Subject = "Paris",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
 });
 return appData;
}

```

```

 {
 Id = 2,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false,
 Status = "Completed",
 Priority = "High"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public string Status { get; set; }
 public string Priority { get; set; }
}

```

### Customize the order of the overlapping events

By default, the scheduler will render the overlapping events based on the start and end time. Now we can customize the order of the overlapping events based on the custom fields by using the **SortComparer** property grouped under the **EventSettings** property. The following code example shows how to sort the appointments based on the custom field as follows.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@{
 Object sortComparer = "sortComparer";
}
<div class="col-lg-9 control-section">
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource, SortComparer = sortComparer })
 .SelectedDate(new DateTime(2017, 9, 29))
 .Render()
)
</div>
<script type="text/javascript">
 function sortComparer (args) {
 return args.sort(function (event1, event2) {
 return event1.RankId.localeCompare(event2.RankId, undefined, {
 numeric: true });
 });
 };
</script>

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
 Subject = "Rank 3",
 StartTime = new DateTime(2023, 9, 29, 10, 30, 0),
 EndTime = new DateTime(2023, 9, 29, 12, 30, 0),
 RankId = "3"
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Rank 1",
 StartTime = new DateTime(2023, 9, 29, 10, 0, 0),
 EndTime = new DateTime(2023, 9, 29, 11, 30),
 RankId = "1"
 });
 eventsData.Add(new EventsData
 {
 Id = 3,
 Subject = "Rank 6",
 StartTime = new DateTime(2023, 9, 29, 7, 0, 0),
 EndTime = new DateTime(2023, 9, 29, 14, 30, 0),
 RankId = "6"
 });
 eventsData.Add(new EventsData
 {
 Id = 4,
 Subject = "Rank 9",
 StartTime = new DateTime(2023, 9, 29, 11, 0, 0),
 EndTime = new DateTime(2023, 9, 29, 15, 30, 0),
 RankId = "9"
 });
 return eventsData;
}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string RankId { get; set; }
}

```

### Drag and drop appointments

Appointments can be rescheduled to any time by dragging and dropping them onto the desired location. To work with drag and drop functionality make sure that `AllowDragAndDrop` is set to **true** on

Scheduler. In mobile mode, you can drag and drop the events by tap holding an event and dropping them on to the desired location.

**Note:** By default, drag and drop action is applicable on all Scheduler views, except Agenda, Month-Agenda and Year view.

#### *Drag and drop multiple appointments*

We can drag and drop multiple appointments by enabling the `allowMultiDrag` property. We can select multiple appointments by holding the CTRL key. Once the events are selected, we can leave the CTRL key and start dragging the event.

We can also drag multiple events from one resource to another resource. In this case, if all the selected events are in the different resources, then all the events should be moved to the single resource that is related to the target event.

**Note:** Multiple events drag and drop is not supported on mobile devices.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .AllowMultiDrag(true)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

*Disable the drag action*

By default, you can drag and drop the events within any of the applicable scheduler views, and to disable it, set **false** to the **AllowDragAndDrop** property.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .AllowDragAndDrop(false)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

*Preventing drag and drop on specific targets*

It is possible to prevent the drag action on particular target, by passing the target to be excluded in the **excludeSelectors** option within **DragStart** event arguments. In this example, we have prevented the drag action on all-day row.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .DragStart("onStart")
```



```

 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
 <script type="text/javascript">
 function onStart(args) {
 args.excludeSelectors = 'e-header-cells,e-header-day,e-header-
date,e-all-day-cells';
 }
 </script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Disable scrolling on drag action

By default, while dragging an appointment to the edges, either top/bottom in the vertical Scheduler or left/right in the timeline Scheduler, scrolling action takes place automatically. To prevent this scrolling, set `false` to the `scroll` value within `DragStart` event arguments.

## CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .DragStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)

```

```
<script type="text/javascript">
 function onStart(args) {
 args.scroll = { enable: false };
 }
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Controlling scroll speed while dragging an event

The speed of the scrolling action while dragging an appointment to the Scheduler edges, can be controlled within **DragStart** event arguments by setting the desired value to the **scrollBy** and **timeDelay** option whereas its default value is 30 minutes and 100 ms.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .DragStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStart(args) {
 args.scroll = { enable: true, scrollBy: 5, timeDelay: 200 };
 }
</script>
```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

*Auto navigation of date ranges on dragging an event*

When an event is dragged either to the left or right extreme edges of the Scheduler and kept hold for few seconds without dropping, the auto navigation of date ranges will be enabled allowing the Scheduler to navigate from current date range to back and forth respectively. This action is set to **false** by default and to enable it, you need to set **navigation** to true within **DragStart** event.

By default, the navigation delay is set to 2000 ms. The navigation delay decides how long the user needs to drag and hold the appointments at the extremities. You can also set your own delay value for letting the users to navigate based on it, using the **timeDelay** within **DragStart** event.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .DragStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStart(args) {
 args.navigation = { enable: true, timeDelay: 4000 };
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### Setting drag time interval

By default, while dragging an appointment, it moves at an interval of 30 minutes. To change the dragging time interval, pass the appropriate values to the `interval` option within `DragStart` event.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .DragStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStart(args) {
 args.interval = 10; // drag interval time is changed to 10 minutes
 }
</script>

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{

```

```

List<AppointmentData> appData = new List<AppointmentData>();
appData.Add(new AppointmentData
{
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
});
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### *Drag and drop items from external source*

It is possible to drag and drop the unplanned items from any of the external source into the scheduler, by manually saving those dropped item as a new appointment data through `addEvent` method of Scheduler.

In this example, we have used the tree view control as an external source and the child nodes from the tree view component are dragged and dropped onto the Scheduler. Therefore, it is necessary to make use of the `nodeDragStop` event of the TreeView component, where we can form an event object and save it using the `addEvent` method.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("500px")
 .Views(view => { view.Option(View.Month).Add(); })
 .SelectedDate(new DateTime(2018, 1, 28))
 .ActionBegin("onActionBegin")
 .Drag("onItemDrag")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .Render()
)
@(Html.EJS().TreeView("tree")
 .Fields(fields =>
fields.DataSource(ViewBag.treeDataSource).Id("Id").Text("Subject")
 .AllowDragAndDrop(true)
 .NodeDragStop("onTreeDragStop")
 .NodeDragging("onItemDrag")
 .Render())
<script type="text/javascript">
 var isTreeItemDropped = false;
 var draggedItemId = '';
 function onItemDrag(event) {
 var scheduleObj = document.querySelector(".e-
schedule").ej2_instances[0];

```

```

 if (scheduleObj.isAdaptive) {
 var classElement = scheduleObj.element.querySelector('.e-device-
hover');
 if (classElement) {
 classElement.classList.remove('e-device-hover');
 }
 if (event.target.classList.contains('e-work-cells')) {
 ej.base.addClass([event.target], 'e-device-hover');
 }
 }
 if (document.body.style.cursor === 'not-allowed') {
 document.body.style.cursor = '';
 }
 if (event.name === 'nodeDragging') {
 var dragElementIcon = document.querySelectorAll('.e-drag-item
.e-icon-expandable');
 for (var i = 0; i < dragElementIcon.length; i++) {
 dragElementIcon[i].style.display = 'none';
 }
 }
 function onActionBegin(event) {
 if (event.requestType === 'eventCreate' && isTreeItemDropped) {
 var treeObj = document.querySelector(".e-
treeview").ej2_instances[0];
 var treeViewdata = treeObj.fields.dataSource;
 var filteredPeople = treeViewdata.filter(function (item) {
return item.Id !== parseInt(draggedItemId, 10); });
 treeObj.fields.dataSource = filteredPeople;
 var elements = document.querySelectorAll('.e-drag-item.treeview-
external-drag');
 for (var i = 0; i < elements.length; i++) {
 remove(elements[i]);
 }
 }
 }
 function onTreeDragStop(event) {
 var treeElement = ej.base.closest(event.target, '.e-treeview');
 var scheduleObj = document.querySelector(".e-
schedule").ej2_instances[0];
 var classElement = scheduleObj.element.querySelector('.e-device-
hover');
 if (classElement) {
 classElement.classList.remove('e-device-hover');
 }
 if (!treeElement) {
 event.cancel = true;
 var scheduleElement = ej.base.closest(event.target, '.e-content-
wrap');
 if (scheduleElement) {
 var treeviewData = this.fields.dataSource;
 if (event.target.classList.contains('e-work-cells')) {
 var filteredData =
 treeviewData.filter(function (item) { return item.Id
=== parseInt(event.draggedNodeData.id, 10); });
 var cellData = scheduleObj.getCellDetails(event.target);

```

```

 var resourceDetails =
scheduleObj.getResourcesByIndex(cellData.groupIndex);
 var eventData = {
 Subject: filteredData[0].Subject,
 StartTime: cellData.startTime,
 EndTime: cellData.endTime
 };
 scheduleObj.addEvent(eventData);
 isTreeItemDropped = true;
 draggedItemId = event.draggedNodeData.id;
 }
}
}
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 ViewBag.treeDataSource = GetTreeData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public List<AppointmentData> GetTreeData()
{
 List<AppointmentData> treeData = new List<AppointmentData>();
 treeData.Add(new AppointmentData
 {
 Id = 11,
 Subject = "Sky Gazers",
 StartTime = new DateTime(2018, 1, 27, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 27, 12, 30, 0)
 });
 treeData.Add(new AppointmentData
 {
 Id = 12,
 Subject = "The Cycle of Seasons",
 StartTime = new DateTime(2018, 1, 28, 14, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 15, 0, 0)
 });
 treeData.Add(new AppointmentData
 {
 Id = 13,

```

```

 Subject = "Aliens vs Humans",
 StartTime = new DateTime(2018, 1, 29, 13, 0, 0),
 EndTime = new DateTime(2018, 1, 29, 14, 0, 0)
 });
 return treeData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### Opening the editor window on drag stop

There are scenarios where you want to open the editor filled with data on newly dropped location and may need to proceed to save it, only when **Save** button is clicked on the editor and on clicking the cancel button should revert these changes. This can be achieved using the **DragStop** event of Scheduler.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .DragStop("onStop")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStop(args) {
 args.cancel = true; //cancels the drop action
 var scheduleObj = document.querySelector(".e-
schedule").ej2_instances[0];
 scheduleObj.openEditor(args.data, "Save"); //open the event window
 with updated start and end time
 }
</script>

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),

```



```

 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Inline Appointment

In Scheduler, another easier way for adding or editing the appointment's subject alone can be achieved by using the inline Add/Edit support. It allows the user to add and edit the appointments inline. To get familiar with the inline Add mode, single click on any of the Scheduler cells or press enter key on the selected cells.

When the inline adding mode is ON, a text box will get created within the clicked Scheduler cells with a blinking cursor in it, requiring the user to enter the subject of an appointment. Once the subject is entered, the appointment will be saved on pressing the enter key.

To enable the inline edit mode, single click on any of the existing appointment's subject, so that the user can edit the subject of that appointment. The edited subject of that appointment will be updated on pressing the enter key.

The inline option can be enabled/disabled on the Scheduler by using the allowInline API, whereas its default value is set to false.

While using the allowInline the showQuickInfo will be turned off. The quickPopup will not show on clicking the work cell or clicking the appointment when the allowInline property is set to true.

In work cells, select multiple cells using keyboard, and then press enter key. The appointment wrapper will be created, and focus will be on the subject field. Also, consider the overlapping scenarios when creating an inline event.

### Normal Event

While editing appointments, single-click the appointment subject, the editable option will be enabled in UI and the cursor will focus at the end of the text. Inline editing will be considered for all possible views.

### Recurrence Event

While editing the occurrence from the recurrence series, it is only possible to edit a single occurrence, not an entire series.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .AllowInline(true)
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

**Appointment Resizing**

Another way of rescheduling an appointment can be done by resizing it through either of its handlers.

To work with resizing functionality make sure that **AllowResizing** property is set to **true**.

*Disable the resize action*

By default, resizing of events is allowed on all Scheduler views except Agenda and Month-Agenda view.

To disable this event resizing action, set false to the **AllowResizing** property.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .AllowResizing(false)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
}

```

```

 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
 }
 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

#### Disable scrolling on resize action

By default, while resizing an appointment, when its handler reaches the extreme edges of the Scheduler, scrolling action will take place along with event resizing. To prevent this scrolling action, set **false** to **scroll** value within the **ResizeStart** event.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .ResizeStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStart(args) {
 args.scroll = { enable: false };
 }
</script>

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData

```

```

 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### *Controlling scroll speed while resizing an event*

The speed of the scrolling action while resizing an appointment to the Scheduler edges, can be controlled within the `ResizeStart` event by setting the desired value to the `scrollBy` option.

### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .ResizeStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStart(args) {
 args.scroll = { enable: true, scrollBy: 5, timeDelay: 200 };
 }
</script>

```

### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}

```

```

}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Setting resize time interval

By default, while resizing an appointment, it extends or shrinks at an interval of 30 minutes. To change this default resize interval, set appropriate values to **interval** option within the **ResizeStart**.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .ResizeStart("onStart")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onStart(args) {
 args.interval = 10; // drag interval time is changed to 10 minutes
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

```
}

```

### Appointment customization

The look and feel of the Scheduler events can be customized using any one of the following ways.

- [Using event template](#)
- [Using EventRendered event](#)
- [Using custom CSS class](#)

#### Using template

Any kind of text, images and links can be added to customize the look of the events. The user can format and change the default appearance of the events by making use of the **Template** option available within the **EventSettings** property. The following code example customizes the appointment's default color and time format.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource, Template = ViewBag.template })
 .Readonly(true)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

<style>
 .e-schedule .e-vertical-view .e-content-wrap .e-appointment {
 border-radius: 8px;
 }
 .e-schedule .e-vertical-view .e-content-wrap .e-appointment .e-
appointment-details {
 padding: 0;
 height: 100%;
 }
 .e-schedule .template-wrap {
 height: 100%;
 white-space: normal;
 }
 .e-schedule .template-wrap .subject {
 font-weight: 600;
 font-size: 15px;
 padding: 4px 4px 4px;
 text-overflow: ellipsis;
 white-space: nowrap;
 overflow: hidden;
 }
 .e-schedule .template-wrap .time {
 height: 50px;
 font-size: 12px;
 padding: 4px 6px 4px;
 overflow: hidden;
 }

```

```

 }
</style>
<script type="text/javascript">
 var instance = new ej.base.Internationalization();
 function getTimeString(value) {
 return instance.formatDate(value, { skeleton: 'hm' });
 }
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetWebinarData();
 ViewBag.template = "<div class='template-wrap'
style='background:${SecondaryColor}'><div class='subject'
style='background:${PrimaryColor}'>${Subject}</div><div class='time'
style='background:${PrimaryColor}'>Time: ${getTimeString(data.StartTime)} -
${getTimeString(data.EndTime)}</div><div class='image'><img src =
'../../Content/schedule/images/${ImageName}.svg'
alt='${ImageName}'/></div><div class='description'>${Description}</div><div
class='footer' style='background:${PrimaryColor}'></div></div>";
 return View();
}

public List<WebinarData> GetWebinarData()
{
 List<WebinarData> webinarData = new List<WebinarData>();
 webinarData.Add(new WebinarData
 {
 Id = 1,
 Subject = "Environment Day",
 Tags = "Eco day, Forest conserving, Earth & its resources",
 Description = "A day that creates awareness to promote the healthy
planet and reduce the air pollution crisis on nature earth.",
 StartTime = new DateTime(2023, 2, 12, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 12, 14, 0, 0),
 ImageName = "environment-day",
 PrimaryColor = "#1aaa55",
 SecondaryColor = "#47bb76"
 });
 webinarData.Add(new WebinarData
 {
 Id = 2,
 Subject = "Health Day",
 Tags = "Reduce mental stress, Follow good food habits",
 Description = "A day that raises awareness on different health
issues. It marks the anniversary of the foundation of WHO.",
 StartTime = new DateTime(2023, 2, 13, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 13, 14, 0, 0),
 ImageName = "health-day",
 PrimaryColor = "#357cd2",
 SecondaryColor = "#5d96db"
 });
 webinarData.Add(new WebinarData
 {
 Id = 3,

```

```

 Subject = "Cancer Day",
 Tags = "Life threatening cancer effects, Palliative care",
 Description = "A day that raises awareness on cancer and its
preventive measures. Early detection saves life.",
 StartTime = new DateTime(2023, 2, 14, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 14, 0, 0),
 ImageName = "cancer-day",
 PrimaryColor = "#7fa900",
 SecondaryColor = "#a4c932"
 });
 webinarData.Add(new WebinarData
 {
 Id = 4,
 Subject = "Happiness Day",
 Tags = "Stress-free, Smile, Resolve frustration and bring
happiness",
 Description = "A general idea is to promote happiness and smile
around the world.",
 StartTime = new DateTime(2023, 2, 15, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 14, 0, 0),
 ImageName = "happiness-day",
 PrimaryColor = "#ea7a57",
 SecondaryColor = "#ee9478"
 });
 webinarData.Add(new WebinarData
 {
 Id = 5,
 Subject = "Tourism Day",
 Tags = "Diverse cultural heritage, strengthen peace",
 Description = "A day that raises awareness on the role of tourism
and its effect on social and economic values.",
 StartTime = new DateTime(2023, 2, 16, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 16, 14, 0, 0),
 ImageName = "tourism-day",
 PrimaryColor = "#00bdae",
 SecondaryColor = "#32cabe"
 });
 return webinarData;
}

public class WebinarData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Tags { get; set; }
 public string Description { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string ImageName { get; set; }
 public string PrimaryColor { get; set; }
 public string SecondaryColor { get; set; }
}

```

**Note:** All the built-in fields that are mapped to the appropriate field properties within the `EventSettings`, as well as custom mapped fields from the Scheduler dataSource can be accessed within the template code.



*Using eventRendered event*

The **EventRendered** event triggers before the appointment renders on the Scheduler. Therefore, this client-side event can be utilized to customize the look of events based on any specific criteria, before rendering them on the scheduler.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventRendered("onEventRendered")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onEventRendered(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 var categoryColor = args.data.CategoryColor;
 if (!args.element || !categoryColor) {
 return;
 }
 if (scheduleObj.currentView === 'Agenda') {
 (args.element.firstChild).style.borderColor = categoryColor;
 } else {
 args.element.style.backgroundColor = categoryColor;
 }
 }
</script>
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2023, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2023, 1, 28, 12, 30, 0),
 CategoryColor = "#357cd2"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
```

```

public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string CategoryColor { get; set; }
}

```

### Using cssClass

The customization of events can also be achieved using `CssClass` property of the Scheduler. In the following example, the background of appointments has been changed using the `cssClass`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .CssClass("custom-class")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource})
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .custom-class.e-schedule .e-vertical-view .e-all-day-appointment-wrapper
.e-appointment,
 .custom-class.e-schedule .e-vertical-view .e-day-wrapper .e-appointment,
 .custom-class.e-schedule .e-month-view .e-appointment{
 background: green;
 }
</style>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetWebinarData();
 return View();
}
public List<WebinarData> GetWebinarData()
{
 List<WebinarData> webinarData = new List<WebinarData>();
 webinarData.Add(new WebinarData
 {
 Id = 1,
 Subject = "Environment Day",
 Tags = "Eco day, Forest conserving, Earth & its resources",
 Description = "A day that creates awareness to promote the healthy planet and reduce the air pollution crisis on nature earth.",
 StartTime = new DateTime(2023, 2, 12, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 12, 14, 0, 0),
 ImageName = "environment-day",
 PrimaryColor = "#1aaa55",
 SecondaryColor = "#47bb76"
 });
 webinarData.Add(new WebinarData

```

```

{
 Id = 2,
 Subject = "Health Day",
 Tags = "Reduce mental stress, Follow good food habits",
 Description = "A day that raises awareness on different health
issues. It marks the anniversary of the foundation of WHO.",
 StartTime = new DateTime(2023, 2, 13, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 13, 14, 0, 0),
 ImageName = "health-day",
 PrimaryColor = "#357cd2",
 SecondaryColor = "#5d96db"
});
webinarData.Add(new WebinarData
{
 Id = 3,
 Subject = "Cancer Day",
 Tags = "Life threatening cancer effects, Palliative care",
 Description = "A day that raises awareness on cancer and its
preventive measures. Early detection saves life.",
 StartTime = new DateTime(2023, 2, 14, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 14, 0, 0),
 ImageName = "cancer-day",
 PrimaryColor = "#7fa900",
 SecondaryColor = "#a4c932"
});
webinarData.Add(new WebinarData
{
 Id = 4,
 Subject = "Happiness Day",
 Tags = "Stress-free, Smile, Resolve frustration and bring
happiness",
 Description = "A general idea is to promote happiness and smile
around the world.",
 StartTime = new DateTime(2023, 2, 15, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 14, 0, 0),
 ImageName = "happiness-day",
 PrimaryColor = "#ea7a57",
 SecondaryColor = "#ee9478"
});
webinarData.Add(new WebinarData
{
 Id = 5,
 Subject = "Tourism Day",
 Tags = "Diverse cultural heritage, strengthen peace",
 Description = "A day that raises awareness on the role of tourism
and its effect on social and economic values.",
 StartTime = new DateTime(2023, 2, 16, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 16, 14, 0, 0),
 ImageName = "tourism-day",
 PrimaryColor = "#00bdae",
 SecondaryColor = "#32cabe"
});
return webinarData;
}
public class WebinarData
{
 public int Id { get; set; }

```

```

public string Subject { get; set; }
public string Tags { get; set; }
public string Description { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string ImageName { get; set; }
public string PrimaryColor { get; set; }
public string SecondaryColor { get; set; }
}

```

### Setting minimum height

It is possible to set minimal height for appointments on Scheduler using `EventRendered` event, when its start and end time duration is less than the default duration of a single slot.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventRendered("onEventRendered")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onEventRendered(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 let cellHeight = (scheduleObj.element.querySelector('.e-work-
cells')).offsetHeight;
 let appHeight = (args.data.EndTime.getTime() -
args.data.StartTime.getTime()) / (60 * 1000) * (36 *
scheduleObj.timeScale.slotCount) / scheduleObj.timeScale.interval;
 args.element.style.height = appHeight + 'px';
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2023, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2023, 1, 28, 12, 30, 0),
 });
}

```

```

 CategoryColor = "#357cd2"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string CategoryColor { get; set; }
}

```

### Block Dates and Times

It is possible to block a set of dates or a particular time ranges on the Scheduler. To do so, define an appointment object within **EventSettings** along with the required time range to block and set the **IsBlock** field to **true**. Usually, the event objects defined with **IsBlock** field set to true will block the entire time cells lying within the appropriate time ranges specified through **StartTime** and **EndTime** fields.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsBlock = true,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
}

```

```

public DateTime EndTime { get; set; }
public bool IsBlock { get; set; }
}

```

Block events can also be defined to repeat on several days as shown in the following code example.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsBlock = true,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=5"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string RecurrenceRule { get; set; }
 public bool IsBlock { get; set; }
}

```

### Readonly

An interaction with the appointments of Scheduler can be enabled/disabled using the **Readonly** property. With this property enabled, you can simply navigate between the Scheduler dates, views and can be able to view the appointment details in the quick info window. Most importantly, the users are not allowed to perform any CRUD actions on Scheduler, when this property is set to true. By default, it is set as **false**.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Readonly(true)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Explosion of Betelgeuse Star",
 StartTime = new DateTime(2018, 2, 15, 9, 30, 0),
 EndTime = new DateTime(2018, 2, 15, 11, 0, 0)
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

**Make specific events readonly**

There are scenarios where you need to restrict the CRUD action on specific appointments alone based on certain conditions. In the following example, the events that has occurred on the past hours from the current date of the Scheduler are made as read-only and the CRUD actions has been prevented only on those appointments. This can be achieved by setting **IsReadonly** field of read-only events to **true**.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
```

```
.Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsReadOnly = true,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsReadOnly { get; set; }
}
```

**Note:** By default, the event editor is prevented to open on the read-only events when `isReadOnly` (for Core) / `IsReadOnly` (for MVC) field is set to `true`.

#### Restricting event creation on specific time slots

You can restrict the users to create and update more than one appointment on specific time slots. Also, you can disable the CRUD action on those time slots if it is already occupied, which can be achieved using Scheduler's public method `isSlotAvailable`.

**Note:** The `isSlotAvailable` is centered around verifying appointments within the present view's date range. Yet, it does not encompass an evaluation of availability for recurrence occurrences that fall beyond this particular date range.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
```



```

 .Render()
)
 <script type="text/javascript">
 function onActionBegin(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 if ((args.requestType === 'eventCreate' || args.requestType ===
'eventChange') && args.data.length > 0) {
 var eventData = args.data[0];
 var eventField = scheduleObj.eventFields;
 var startDate = eventData[eventField.startTime];
 var endDate = eventData[eventField.endTime];
 args.cancel = !scheduleObj.isSlotAvailable(startDate, endDate);
 }
 }
 </script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Meteor Showers in 2018", StartTime = new
DateTime(2018, 2, 14, 13, 0, 0), EndTime = new DateTime(2018, 2, 14, 14, 30,
0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 12, 0, 0), EndTime = new DateTime(2018, 2, 15, 14, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Differentiate the past time events

To differentiate the appearance of the appointments based on specific criteria such as displaying the past hour appointments with different colors on Scheduler, `EventRendered` event can be used which triggers before the appointment renders on the Scheduler.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventRendered("onEventRendered")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 29))
 .Render()
)
<style>
 .e-past-app {
 background-color: chocolate !important;
 }
</style>
<script type="text/javascript">
 function onEventRendered(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 if (args.data.EndTime < scheduleObj.selectedDate) {
 args.element.classList.add('e-past-app');
 }
 }
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2018, 1, 28, 10, 0, 0),
 EndTime = new DateTime(2018, 1, 28, 12, 30, 0),
 CategoryColor = "#357cd2"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
```

```

public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string CategoryColor { get; set; }
}

```

### Appointments occupying entire cell

The Scheduler allows the event to occupies the full height of the cell without its header part by setting true for **EnableMaxHeight** Property.

We can show more indicator if more than one appointment is available in a same cell by setting true to **EnableIndicator** property whereas its default value is false.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource, EnableMaxHeight = true, EnableIndicator = false })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
 Subject = "Server Maintenance",
 StartTime = new DateTime(2023, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 11, 30, 0),
 EventType = "maintenance",
 City = "Seattle",
 CategoryColor = "#1aaa55"
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Art & Painting Gallery",
 StartTime = new DateTime(2023, 2, 15, 12, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 14, 0, 0),
 EventType = "public-event",
 City = "Costa Rica",
 CategoryColor = "#357cd2"
 });
 eventsData.Add(new EventsData

```

```

 {
 Id = 3,
 Subject = "Dany Birthday Celebration",
 StartTime = new DateTime(2023, 2, 16, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 16, 11, 30, 0),
 EventType = "family-event",
 City = "Kirkland",
 CategoryColor = "#7fa900"
 });
 return eventsData;
}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string EventType { get; set; }
 public string City { get; set; }
 public string CategoryColor { get; set; }
}

```

#### How to limit maximum number of events to display

In the Scheduler, the default behavior is to display concurrent events based on cell height, with each new event represented as **+n more** characters. However, you may want to improve the quality of the presentation by limiting the number of concurrent events. This can be accomplished by using the `maxEventsPerRow` property, which is defaulted to the `views` property.

The `maxEventsPerRow` property is specific to the month, timeline month, and timeline year views, allowing you to view events visually in these rows. Below is a code example that demonstrates how to use this constraint and the events displayed in a cell have been created:

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Month).MaxEventsPerRow(3).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2023, 11, 10))
 .Render()
)

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()

```

```

{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {Id: 1, Subject: 'Explosion of Betelgeuse Star', StartTime: new
Date(2023, 11, 11, 9, 30), EndTime: new Date(2023, 11, 11, 11, 0) });
 appData.Add(new AppointmentData
 {Id: 2, Subject: 'Thule Air Crash Report', StartTime: new Date(2023, 11,
11, 12, 30), EndTime: new Date(2023, 11, 11, 13, 0) });
 appData.Add(new AppointmentData
 {Id: 3, Subject: 'Blue Moon Eclipse', StartTime: new Date(2023, 11, 11,
13, 30), EndTime: new Date(2023, 11, 11, 15, 30) });
 appData.Add(new AppointmentData
 {Id: 4, Subject: 'Meteor Showers in 2023', StartTime: new Date(2023, 11,
14, 13, 0), EndTime: new Date(2023, 11, 14, 14, 30) });
 appData.Add(new AppointmentData
 {Id: 5, Subject: 'Milky Way as Melting pot', StartTime: new Date(2023,
11, 14, 15, 0), EndTime: new Date(2023, 11, 14, 15, 30) });
 appData.Add(new AppointmentData
 {Id: 6, Subject: 'Mysteries of Bermuda Triangle', StartTime: new
Date(2023, 11, 15, 9, 30), EndTime: new Date(2023, 11, 15, 11, 0) });
 appData.Add(new AppointmentData
 {Id: 7, Subject: 'Glaciers and Snowflakes', StartTime: new Date(2023,
11, 15, 11, 0), EndTime: new Date(2023, 11, 15, 12, 30) });
 appData.Add(new AppointmentData
 {Id: 8, Subject: 'Life on Mars', StartTime: new Date(2023, 11, 17, 9,
0), EndTime: new Date(2023, 11, 17, 10, 0) });
 appData.Add(new AppointmentData
 {Id: 9, Subject: 'Alien Civilization', StartTime: new Date(2023, 11, 17,
12, 0), EndTime: new Date(2023, 11, 17, 13, 0) });
 appData.Add(new AppointmentData
 {Id: 10, Subject: 'Wildlife Galleries', StartTime: new Date(2023, 11,
17, 15, 0), EndTime: new Date(2023, 11, 17, 17, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

The property `maxEventsPerRow` will be applicable only when [rowAutoHeight](#) feature is disabled in the Scheduler.

| Sunday | Monday                                                                                            | Tuesday                                                                                           | Wednesday                                                                                | Thursday                                                                                       | Friday                                                                                                  | Saturday |
|--------|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|----------|
| 16     | 17<br>2:18 AM Meet a small Ma...<br>2:36 AM Venomous Snake...<br>8:31 AM Amazon Fish Fee...       | 18<br>1:31 AM Walk with Jungle...                                                                 | 19<br>10:02 AM Croco World<br>3:24 PM Parrot Talk<br>7:30 PM Wildlife Warriors           | 20<br>3:31 PM Croco World                                                                      | 21<br>4:08 AM Feed the Giants<br>2:58 PM Pony Rides<br>Feed the Giants                                  | 22       |
| 23     | 24<br>3:34 AM Camping with Tu...<br>2:32 PM Wildlife Warriors<br>4:45 PM Birds of Prey<br>+1 more | 25<br>1:40 AM Face Painting & ...<br>3:41 PM Feed the Giants<br>7:23 PM Parrot Talk               | 26<br>Parrot Talk                                                                        | 27<br>7:30 AM Meet a small Ma...<br>7:02 PM Face Painting & ...<br>2:15 PM Face Painting & ... | 28<br>7:04 AM Endangered Spe...<br>1:39 PM Pony Rides<br>8:12 PM Parrot Talk                            | 29       |
| 30     | 31<br>1:01 AM Pony Rides<br>8:34 AM Venomous Snake...                                             | Feb 1<br>Camping with Turtles<br>2:15 AM Venomous Snake...<br>12:18 PM Feed the Giants<br>+1 more | 2<br>11:32 AM Croco World<br>12:28 PM Playtime with C...<br>2:53 PM Story Time for Ki... | 3<br>1:15 AM Camping with Tu...<br>5:31 AM Croco World                                         | 4<br>12:28 PM Camping with T...<br>5:48 PM Camping with Tu...<br>7:44 PM Face Painting & ...<br>+1 more | 5        |
| 6      | 7<br>1:16 AM Birds of Prey<br>8:34 AM Amazon Fish Fee...<br>11:27 AM Venomous Snake...            | 8<br>1:55 AM Pony Rides<br>6:42 AM Face Painting & ...<br>7:37 PM Camping with Tu...              | 9<br>Story Time for Kids<br>9:03 AM Camping with Tu...<br>11:35 AM Black Cockatoos...    | 10<br>10:39 AM Venomous Snake...<br>7:29 PM Wildlife Warriors<br>8:38 PM Birds of Prey         | 11                                                                                                      | 12       |

### Display tooltip for appointments

The tooltip shows the Scheduler appointment's information in a formatted style by making use of the tooltip related options.

#### Show or hide built-in tooltip

The tooltip can be displayed for appointments by setting `true` to the `EnableTooltip` option within the `EventSettings` property.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource, EnableTooltip = true })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
```

```

 Subject = "Server Maintenance",
 StartTime = new DateTime(2023, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 11, 30, 0),
 EventType = "maintenance",
 City = "Seattle",
 CategoryColor = "#1aaa55"
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Art & Painting Gallery",
 StartTime = new DateTime(2023, 2, 15, 12, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 14, 0, 0),
 EventType = "public-event",
 City = "Costa Rica",
 CategoryColor = "#357cd2"
 });
 eventsData.Add(new EventsData
 {
 Id = 3,
 Subject = "Dany Birthday Celebration",
 StartTime = new DateTime(2023, 2, 16, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 16, 11, 30, 0),
 EventType = "family-event",
 City = "Kirkland",
 CategoryColor = "#7fa900"
 });
 return eventsData;
}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string EventType { get; set; }
 public string City { get; set; }
 public string CategoryColor { get; set; }
}

```

### Customizing event tooltip using template

After enabling the default tooltip, it is possible to customize the display of needed event information on tooltip by making use of the `TooltipTemplate` option within the `EventSettings`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@{
 var template = "<div class='tooltip-wrap'>" +
 "<div class='content-area'><div class='name'>${Subject}</div> " +
 "${if(City != null && City != undefined)}<div"
 class='city'>${City}</div>${if}</div> " +
 "<div class='time'>From : ${StartTime.toLocaleString()} </div> " +
 "<div class='time'>To : ${EndTime.toLocaleString()} "
 </div></div></div>";
}

```

```

@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings {
 DataSource = ViewBag.datasource,
 EnableTooltip = true,
 TooltipTemplate = template
 })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
 Subject = "Server Maintenance",
 StartTime = new DateTime(2023, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 11, 30, 0),
 EventType = "maintenance",
 City = "Seattle",
 CategoryColor = "#1aaa55"
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Art & Painting Gallery",
 StartTime = new DateTime(2023, 2, 15, 12, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 14, 0, 0),
 EventType = "public-event",
 City = "Costa Rica",
 CategoryColor = "#357cd2"
 });
 eventsData.Add(new EventsData
 {
 Id = 3,
 Subject = "Dany Birthday Celebration",
 StartTime = new DateTime(2023, 2, 16, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 16, 11, 30, 0),
 EventType = "family-event",
 City = "Kirkland",
 CategoryColor = "#7fa900"
 });
 return eventsData;
}
public class EventsData
{

```



```

public int Id { get; set; }
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string EventType { get; set; }
public string City { get; set; }
public string CategoryColor { get; set; }
}

```

**Note:** All the field names that are mapped from the Scheduler dataSource to the appropriate field properties such as subject, description, location, startTime and endTime within the `EventSettings` can be accessed within the template.

### Appointment selection

Appointment selection can be done either through mouse or keyboard actions. The selected events in UI will have a box shadow effect around to differentiate it from other appointments.

| Action                                               | Description                    |
|------------------------------------------------------|--------------------------------|
| ----- -----                                          |                                |
| Mouse click or Single tap on appointments            | Selects single appointment.    |
| Ctrl + [Mouse click] or [Single tap] on appointments | Selects multiple appointments. |

### Deleting multiple appointments

With the options available to select multiple appointments, it is also possible to delete the multiple selected appointments simply by pressing the `Delete` key. In case of deleting multiple selected occurrences of an event series, only those occurrences will be deleted and not the entire series.

### Retrieve event details from the UI of an event

It is possible to access the information about the event fields of an appointment element displayed on the Scheduler UI. This can be achieved by passing an appointment element as argument to the public method `getEventDetails`.

In the following example, the subject of the appointment clicked has been displayed.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
<div class="col-lg-9 control-section">
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventClick("onEventClick")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<div class="col-lg-3 property-section">
 <table id="property" title="Event Trace">
 <tbody>
 <tr>
 <td>

```

```

 <div class="eventarea" style="height: 245px;overflow:
auto">
 <span class="EventLog" id="EventLog" style="word-
break: normal;">
 </div>
 </td>
</tr>
<tr>
 <td>
 <div class="eventbtn" style="padding-bottom: 10px">
 <input id="clear" type="button" class="btn btn-
default" value="Clear">
 </div>
 </td>
</tr>
</tbody>
</table>
</div>
<style>
 #EventLog b {
 color: #388e3c;
 }
 hr {
 margin: 1px 10px 1px 0px;
 border-top: 1px solid #eee;
 }
</style>
<script type="text/javascript">
 document.querySelector('.eventbtn').onclick = function () {
 document.querySelector('.EventLog').innerHTML = '';
 };
 function onEventClick(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 var event = scheduleObj.getEventDetails(args.element);
 appendElement(event.Subject + '<hr>');
 }
 function appendElement(html) {
 var span = document.createElement('span');
 span.innerHTML = html;
 var log = document.querySelector('.EventLog');
 log.insertBefore(span, log.firstChild);
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData

```

```

{
 Id = 1,
 Subject = "Server Maintenance",
 StartTime = new DateTime(2018, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 14, 11, 30, 0),
 EventType = "maintenance",
 City = "Seattle",
 CategoryColor = "#1aaa55"
});
eventsData.Add(new EventsData
{
 Id = 2,
 Subject = "Art & Painting Gallery",
 StartTime = new DateTime(2018, 2, 15, 12, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 14, 0, 0),
 EventType = "public-event",
 City = "Costa Rica",
 CategoryColor = "#357cd2"
});
eventsData.Add(new EventsData
{
 Id = 3,
 Subject = "Dany Birthday Celebration",
 StartTime = new DateTime(2018, 2, 16, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 16, 11, 30, 0),
 EventType = "family-event",
 City = "Kirkland",
 CategoryColor = "#7fa900"
});
return eventsData;
}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string EventType { get; set; }
 public string City { get; set; }
 public string CategoryColor { get; set; }
}

```

### Get the current view appointments

To retrieve the appointments present in the current view of the Scheduler, you can make use of the `getCurrentViewEvents` public method. In the following example, the count of current view appointment collection rendered has been traced in `DataBound` event.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
<div class="col-lg-9 control-section">
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .DataBound("onDataBound")

```

```

 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<div class="col-lg-3 property-section">
 <table id="property" title="Event Trace">
 <tbody>
 <tr>
 <td>
 <div class="eventarea" style="height: 245px;overflow:
auto">
 <span class="EventLog" id="EventLog" style="word-
break: normal;">
 </div>
 </td>
 </tr>
 <tr>
 <td>
 <div class="evtbtn" style="padding-bottom: 10px">
 <input id="clear" type="button" class="btn btn-
default" value="Clear">
 </div>
 </td>
 </tr>
 </tbody>
 </table>
</div>
<style>
 #EventLog b {
 color: #388e3c;
 }
 hr {
 margin: 1px 10px 1px 0px;
 border-top: 1px solid #eee;
 }
</style>
<script type="text/javascript">
 document.getElementById('clear').onclick = function () {
 document.getElementById('EventLog').innerHTML = '';
 };
 function onDataBound(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var event = scheduleObj.getCurrentViewEvents();
 if (event.length > 0) {
 appendElement('Events present on current view ' +
event.length + '<hr>');
 } else {
 appendElement('No Events available in this view.<hr>');
 }
 }
 function appendElement(html) {
 var span = document.createElement('span');
 span.innerHTML = html;
 var log = document.getElementById('EventLog');

```

```

 log.insertBefore(span, log.firstChild);
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
 Subject = "Server Maintenance",
 StartTime = new DateTime(2018, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 14, 11, 30, 0),
 EventType = "maintenance",
 City = "Seattle",
 CategoryColor = "#1aaa55"
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Art & Painting Gallery",
 StartTime = new DateTime(2018, 2, 15, 12, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 14, 0, 0),
 EventType = "public-event",
 City = "Costa Rica",
 CategoryColor = "#357cd2"
 });
 eventsData.Add(new EventsData
 {
 Id = 3,
 Subject = "Dany Birthday Celebration",
 StartTime = new DateTime(2018, 2, 16, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 16, 11, 30, 0),
 EventType = "family-event",
 City = "Kirkland",
 CategoryColor = "#7fa900"
 });
 return eventsData;
}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string EventType { get; set; }
 public string City { get; set; }
 public string CategoryColor { get; set; }
}

```

### Get the entire appointment collections

The entire collection of appointments rendered on the Scheduler can be accessed using the `getEvents` public method. In the following example, the count of entire appointment collection rendered on the Scheduler has been traced in `DataBound` event.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
<div class="col-lg-9 control-section">
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .DataBound("onDataBound")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<div class="col-lg-3 property-section">
 <table id="property" title="Event Trace">
 <tbody>
 <tr>
 <td>
 <div class="eventarea" style="height: 245px;overflow:
auto">
 <span class="EventLog" id="EventLog" style="word-
break: normal;">
 </div>
 </td>
 </tr>
 <tr>
 <td>
 <div class="evtbtn" style="padding-bottom: 10px">
 <input id="clear" type="button" class="btn btn-
default" value="Clear">
 </div>
 </td>
 </tr>
 </tbody>
 </table>
</div>
<style>
 #EventLog b {
 color: #388e3c;
 }
 hr {
 margin: 1px 10px 1px 0px;
 border-top: 1px solid #eee;
 }
</style>
<script type="text/javascript">
 document.getElementById('clear').onclick = function () {
 document.getElementById('EventLog').innerHTML = '';
 };
</script>
```

```

function onDataBound(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var event = scheduleObj.getEvents();
 appendElement('Events present on scheduler ' + event.length
+ '<hr>');
}
function appendElement(html) {
 var span = document.createElement('span');
 span.innerHTML = html;
 var log = document.getElementById('EventLog');
 log.insertBefore(span, log.firstChild);
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 return View();
}
public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
 Subject = "Server Maintenance",
 StartTime = new DateTime(2018, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 14, 11, 30, 0),
 EventType = "maintenance",
 City = "Seattle",
 CategoryColor = "#1aaa55"
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Art & Painting Gallery",
 StartTime = new DateTime(2018, 2, 15, 12, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 14, 0, 0),
 EventType = "public-event",
 City = "Costa Rica",
 CategoryColor = "#357cd2"
 });
 eventsData.Add(new EventsData
 {
 Id = 3,
 Subject = "Dany Birthday Celebration",
 StartTime = new DateTime(2018, 2, 16, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 16, 11, 30, 0),
 EventType = "family-event",
 City = "Kirkland",
 CategoryColor = "#7fa900"
 });
 return eventsData;
}

```

```

}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string EventType { get; set; }
 public string City { get; set; }
 public string CategoryColor { get; set; }
}

```

### Refresh appointments

If your requirement is to simply refresh the appointments instead of refreshing the entire Scheduler elements from your application end, make use of the `refreshEvents` public method.

```
`sh
```

```
scheduleObj.refreshEvents();
```

```
,
```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Data-binding

The Scheduler uses `DataManager`, which supports both RESTful JSON data services binding and local JavaScript object array binding. The `DataSource` property can be assigned either with the instance of `DataManager` or JavaScript object array collection. It supports two kinds of data binding method:

- Local data
- Remote data

### Binding local data

To bind local JSON data to the Scheduler, you can simply assign a JavaScript object array to the `DataSource` option of the scheduler within the `EventSettings` property. The local data source can also be provided as an instance of the `DataManager`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS



```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 12, 9, 30, 0), EndTime = new DateTime(2018, 2, 12, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 14, 9, 30, 0), EndTime = new DateTime(2018, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** By default, `DataManager` uses `JsonAdaptor` for local data-binding.

You can also bind different field names to the default event fields as well as include additional custom fields to the event object collection which can be referred [here](#).

#### Binding remote data

Any kind of remote data services can be bound to the Scheduler. To do so, create an instance of `DataManager` and provide the service URL to the `Url` option of `DataManager` and then assign it to the `DataSource` property within `EventSettings`.

#### Using ODataV4Adaptor

[ODataV4](#) is a standardized protocol for creating and consuming data. Refer to the following code example to retrieve the data from ODataV4 service using the `DataManager`. To connect with ODataV4 service end points, it is necessary to make use of `ODataV4Adaptor` within `DataManager`.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Readonly(true)
 .EventSettings(e =>
 e.DataSource(d =>
 d.Url("https://ej2services.syncfusion.com/production/web-
services/api/Schedule")
 .Adaptor("ODataV4Adaptor")
 .CrossDomain(true)
)
)
)

```

```

)
 .SelectedDate(new DateTime(2020, 9, 20))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

#### *Filter events using the in-built query*

To enable server-side filtering operations based on predetermined conditions, the [includeFiltersInQuery](#) API can be set to true, this allows the filter query to be constructed using the start date, end date, and recurrence rule which in turn enables the request to be filtered accordingly.

This method greatly improves the component's performance by reducing the data that needs to be transferred to the client side. As a result, the component's efficiency and responsiveness are significantly enhanced, resulting in a better user experience. However, it is important to consider the possibility of longer query strings, which may cause issues with the maximum URL length or server limitations on query string length.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Readonly(true)
 .EventSettings(e =>
 e.DataSource(d =>

 d.Url("https://services.odata.org/V4/Northwind/Northwind.svc/Orders/")
 .Adaptor("ODataV4Adaptor")
 .CrossDomain(true)
)
 .Fields(f =>
 f.Subject(sub => sub.Name("ShipName"))
 .Id("Id")
 .Location(loc => loc.Name("ShipCountry"))
 .Description(des => des.Name("ShipAddress"))
 .StartTime(st => st.Name("OrderDate"))
 .EndTime(et => et.Name("RequiredDate"))
 .RecurrenceRule(rec => rec.Name("ShipRegion"))
)
 .Query("new ej.data.Query()")
 .IncludeFiltersInQuery(true)
)
 .SelectedDate(new DateTime(1996, 7, 9))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}
public class AppointmentData
{
 public int Id { get; set; }
 public string ShipName { get; set; }
 public DateTime OrderDate { get; set; }
 public DateTime RequiredDate { get; set; }
 public string ShipCountry { get; set; }
 public string ShipAddress { get; set; }
 public string ShipRegion { get; set; }
}

```

The following image represents how the parameters are passed using ODataV4 filter.

**Request URL:** https://localhost:44360/api/Events?\$filter=(((Start Time%20ge%202022-11-26T18:30:00.000Z)%20and%20(EndTime%20ge%202022-11-26T18:30:00.000Z))%20and%20(StartTime%20lt%202022-12-31T18:30:00.000Z))%20or%20((StartTime%20le%202022-11-26T18:30:00.000Z)%20and%20(EndTime%20gt%202022-11-26T18:30:00.000Z))%20or%20((RecurrenceRule%20ne%20null)%20and%20(RecurrenceRule%20ne%20%27%27))&readOnly=true

**Request Method:** GET

**Status Code:** 200

**Remote Address:** [::1]:44360

**Referrer Policy:** strict-origin-when-cross-origin

#### Using custom adaptor

It is possible to create your own custom adaptor by extending the built-in available adaptors. The following example demonstrates the custom adaptor usage and how to add a custom field **EventID** for the appointments by overriding the built-in response processing using the **processResponse** method of the **ODataV4Adaptor**.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Readonly(true)

```

```

 .SelectedDate(new DateTime(2020, 9, 20))
 .Created("created")
 .Render()
)
 <script type="text/javascript">
 function created(args) {
 class CustomAdaptor extends ej.data.ODataV4Adaptor {
 processResponse() {
 var i = 0;
 // calling base class processResponse function
 var original = super.processResponse.apply(this, arguments);
 // adding employee id
 original.forEach(function (item) { item['EmpID'] = ++i });
 return original;
 }
 }
 var dataManager = new ej.data.DataManager({
 url: 'https://ej2services.syncfusion.com/production/web-
services/api/Schedule',
 adaptor: new CustomAdaptor
 });
 var schObj = document.querySelector('.e-schedule').ej2_instances[0];
 schObj.eventSettings.dataSource = dataManager;
 }
 </script>

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

### Loading data via AJAX post

You can bind the event data through external ajax request and assign it to the **DataSource** property of Scheduler. In the following code example, we have retrieved the data from server with the help of ajax request and assigned the resultant data to the **DataSource** property of Scheduler within the **onSuccess** event of Ajax.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
<input type="button" value="Click me to load Event" onclick="GetData()" />

@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .ReadOnly(true)
 .SelectedDate(new DateTime(2017, 6, 11))
 .Render()
)
<script type="text/javascript">
 function GetData() {
 var ajax = new ej.base.Ajax('/Home/GetData', 'Post', false);
 ajax.send().then(

```

```

 function (value) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 scheduleObj.eventSettings.dataSource = JSON.parse(value);
 scheduleObj.dataBind();
 },
 function (reason) {
 console.log(reason);
 });
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

**Note:** Definition for the controller method `GetData` can be referred [here](#).

### Passing additional parameters to the server

To send an additional custom parameter to the server-side post, you need to make use of the `addParams` method of `Query`. Now, assign this `Query` object with additional parameters to the `Query` property of `Scheduler`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Readonly(true)
 .EventSettings(e =>
 e.DataSource(d =>
 d.Url("https://ej2services.syncfusion.com/development/web-services/odata/")
 .Adaptor("ODataV4Adaptor")
 .CrossDomain(true)
)
 .Query("new ej.data.Query().from('Events').addParams('readOnly',
'true')")
)
 .SelectedDate(new DateTime(2017, 6, 11))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

**Note:** The parameters added using the **Query** property will be sent along with the data request sent to the server on every scheduler actions.

### Handling failure actions

During the time of Scheduler interacting with server, there are chances that some server-side exceptions may occur. You can acquire those error messages or exception details in client-side using the [ActionFailure](#) event of Scheduler.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(e =>
 e.DataSource(d =>
 d.Url("http://some.com/invalidUrl").Adaptor("UrlAdaptor").CrossDomain(true)
)
).ActionFailure("onActionFailure")
 .SelectedDate(new DateTime(2017, 6, 11))
 .Render()
)
<script type="text/javascript">
 function onActionFailure(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var span = document.createElement('span');
 scheduleObj.element.parentNode.insertBefore(span,
scheduleObj.element);
 span.style.color = '#FF0000';
 span.innerHTML = 'Server exception: 404 Not found';
 }
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 return View();
}
```

The argument passed to the **ActionFailure** event contains the error details returned from the server.

### Scheduler CRUD actions

The CRUD (Create, Read, Update and Delete) actions can be performed easily on Scheduler appointments using the various adaptors available within the **DataManager**. Most preferably, we will be using **UrlAdaptor** for performing CRUD actions on scheduler appointments.

```
`sh
```

```
@(Html.EJS().Schedule("schedule")
.Width("100%")
```

```

.Height("550px")
.EventSettings(e => e.DataSource(d =>
d.Url("Home/GetData").CrudUrl("Home/UpdateData").Adaptor("UrlAdaptor").CrossDomain(true)))
.SelectedDate(new DateTime(2017, 6, 5))
.Render()
)
,

```

The server-side controller code to handle the CRUD operations are as follows.

```

`sh
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using ScheduleSample.Models;
namespace ScheduleSample.Controllers
{
public class HomeController : Controller
{
ScheduleDataDataContext db = new ScheduleDataDataContext();
public ActionResult Index()
{
return View();
}

public ActionResult LoadData() // Here we get the Start and End Date and based on that can filter the
data and return to Scheduler
{
var data = db.ScheduleEventDatas.ToList();
return Json(data);
}

[HttpPost]
public ActionResult UpdateData([FromBody] EditParams param)
{

```

```
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
Id = intMax + 1,
StartTime = startTime,
EndTime = endTime,
Subject = value.Subject,
IsAllDay = value.IsAllDay,
StartTimezone = value.StartTimezone,
EndTimezone = value.EndTimezone,
RecurrenceRule = value.RecurrenceRule,
RecurrenceID = value.RecurrenceID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
```



```
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime;
appointment.EndTime = endTime;
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
var data = db.ScheduleEventDatas.ToList();
```

```

return Json(data);
}

public class EditParams
{
 public string key { get; set; }
 public string action { get; set; }
 public List<ScheduleEventData> added { get; set; }
 public List<ScheduleEventData> changed { get; set; }
 public List<ScheduleEventData> deleted { get; set; }
 public ScheduleEventData value { get; set; }
}
}
}
,

```

### Configuring Scheduler with Google API service

We have assigned our custom created Google Calendar url to the DataManager and assigned the same to the Scheduler **DataSource**. Since the events data retrieved from the Google Calendar will be in its own object format, therefore it needs to be resolved manually within the Scheduler's **DataBinding** event. Within this event, the event fields needs to be mapped properly and then assigned to the result.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Schedule
@{
 var calendarId = "5105trob9dasha31vuqek6qgp0@group.calendar.google.com";
 var publicKey = "AIzaSyD76zjMDsL_jkenM5AAnNsORypS1Icuqyg";
 var dataManager = new DataManager() { Url =
"https://www.googleapis.com/calendar/v3/calendars/" + calendarId +
"/events?key=" + publicKey, Adaptor = "WebApiAdaptor", CrossDomain = true };
}
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .EventSettings(new ScheduleEventSettings { DataSource = dataManager })
 .DataBinding("onDataBinding")
 .Readonly(true)
 .SelectedDate(new DateTime(2018, 11, 14))
 .Render()
)
<script type="text/javascript">
 function onDataBinding(e) {
 var items = e.result.items;
 var scheduleData = [];
 if (items.length > 0) {
 for (var i = 0; i < items.length; i++) {

```

```

 var event = items[i];
 var when = event.start.dateTime;
 var start = event.start.dateTime;
 var end = event.end.dateTime;
 if (!when) {
 when = event.start.date;
 start = event.start.date;
 end = event.end.date;
 }
 scheduleData.push({
 Id: event.id,
 Subject: event.summary,
 StartTime: new Date(start),
 EndTime: new Date(end),
 IsAllDay: !event.start.dateTime
 });
 }
 e.result = scheduleData;
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### CRUD Actions

Events, a.k.a. Appointments, play an important role in Scheduler with which the users mostly interact. You can easily manipulate (add/edit/delete) the desired appointments as and when required either using the editor window or through the drag and resize action.

#### Add

Any kind of appointments such as normal, all-day, spanned or recurring events can be easily added on Scheduler using any one of the following ways.

- [Creation using editor window](#)
- [Creation using addEvent method](#)

#### Creation using editor window

The default editor window opens when you double click on the Scheduler cells. It provides you with event related options such as Subject, Location, Start and End time, All-day, Timezone, Description and other recurrence options. With these available fields, you can choose to provide detailed information to the events. Once the fields are filled with proper values, enter the **Save** button to add an event.

In case, if you want to simply provide the Subject alone for appointments, just single click on the required cells which will open the quick popup expecting you to enter subject alone and save it. You can also select multiple cells and press **Enter** key to open the quick popup for selected time range and save the appointment for that time range.

In case, if you need to add some other additional fields to the editor window, then you can opt for [custom editor window](#) which allows you to include fields as per your application needs. If you need to add just one or two [additional fields to the existing default editor window](#), you can do so by defining it manually and then appending it to the editor window.

#### *Creation using addEvent method*

The appointments can be created dynamically by using **addEvent** method. Either you can add a single or a collection of appointment objects using **addEvent** method. The following code example let you know how to use the **addEvent** method to create multiple appointments simultaneously.

#### **CSHTML**

```
@using Syncfusion.EJ2.Schedule
@using Syncfusion.EJ2.Buttons
@ (Html.EJS().Button("btn1")
 .Content("ADD")
 .Render()
)
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var data = [{
 Id: 1,
 Subject: 'Conference',
 StartTime: new Date(2018, 1, 12, 9, 0),
 EndTime: new Date(2018, 1, 12, 10, 0),
 IsAllDay: false
 }, {
 Id: 2,
 Subject: 'Meeting',
 StartTime: new Date(2018, 1, 15, 10, 0),
 EndTime: new Date(2018, 1, 15, 11, 30),
 IsAllDay: false
 }
];
 scheduleObj.addEvent(data);
 };
</script>
```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 3,
 Subject = "Testing",
 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
 });
 appData.Add(new AppointmentData
 {
 Id = 4,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 13, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 13, 10, 0, 0),
 IsAllDay = false,
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

#### Inserting events into database at server-side

While adding the normal or recurring events to the Scheduler, **insert** action takes place and the following code example describes how to add a new event into database at server side.

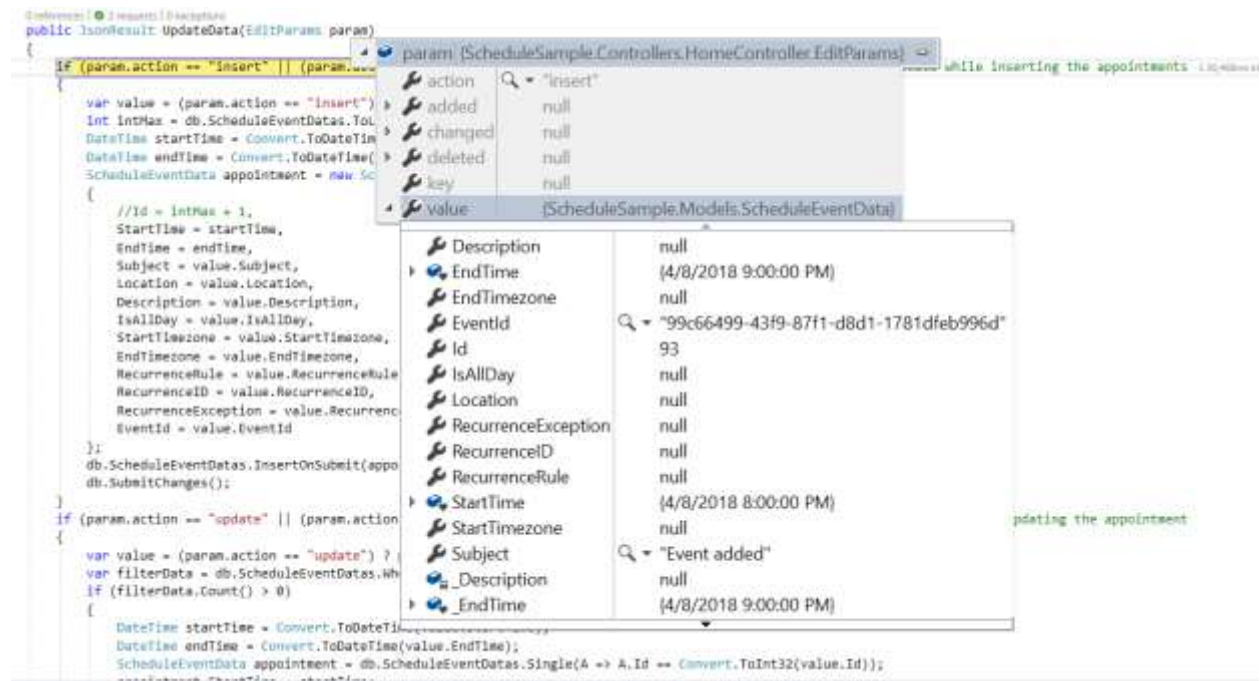
```
`sh
```

```
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
```

```
{
```

```
var value = (param.action == "insert") ? param.value : param.added[0];
```

```
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
1;
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
 Id = intMax + 1,
 StartTime = startTime,
 EndTime = endTime,
 Subject = value.Subject,
 IsAllDay = value.IsAllDay,
 StartTimezone = value.StartTimezone,
 EndTimezone = value.EndTimezone,
 RecurrenceRule = value.RecurrenceRule,
 RecurrenceID = value.RecurrenceID,
 RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}
```



### Restricting add action based on specific criteria

In the following example, the specific fields of Scheduler editor window such as Subject and Location are made to undergo validation such that if it is left as blank, then the default **required** validation message will be displayed, while clicking on a save button.

Additionally, the regex condition has been added to the Location field, so that if any special characters are typed into it, then the custom validation message will be displayed.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@{
 var validationRules = new Dictionary<string, object>() { { "required",
true } };
 var locationValidationRules = new Dictionary<string, object>() { {
"required", true }, { "regex", new string[] { "^[a-zA-Z0-9-]*$", "Special
character(s) not allowed in this field" } } };
 var descriptionValidationRules = new Dictionary<string, object>() { {
"required", true }, { "minLength", 5 }, { "maxLength", 500 } };
}
@(Html.EJS().Schedule("schedule")
.Width("100%")
.Height("650px")
.EventSettings(e => e.Fields(f => f.Subject(sub =>
sub.Name("Subject").Validation(validationRules))
.Location(loc =>
loc.Name("Location").Validation(locationValidationRules))
.Description(des =>
des.Name("Description").Validation(descriptionValidationRules))
.StartTime(st =>
st.Name("StartTime").Validation(validationRules))
.EndTime(et => et.Name("EndTime").Validation(validationRules))
)
.DataSource(ViewBag.datasources))
```

```

)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Meteor Showers in 2018", StartTime = new
DateTime(2018, 2, 14, 13, 0, 0), EndTime = new DateTime(2018, 2, 14, 14, 30,
0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 12, 0, 0), EndTime = new DateTime(2018, 2, 15, 14, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

You can also dynamically prevent the creation of appointments on Scheduler. For example, say if you want to decline the creation of appointments on weekend days, you can check for its appropriate condition within the **ActionBegin** event.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
)

```



```

 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onActionBegin(args) {
 var weekEnds = [0, 6];
 if ((args.requestType === 'eventCreate' &&
weekEnds.indexOf((args.data[0].StartTime).getDay()) >= 0)) {
 args.cancel = true;
 }
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Meteor Showers in 2018", StartTime = new
DateTime(2018, 2, 14, 13, 0, 0), EndTime = new DateTime(2018, 2, 14, 14, 30,
0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 12, 0, 0), EndTime = new DateTime(2018, 2, 15, 14, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

## Edit

The same way the appointments such as normal, all-day, spanned or recurring events are created, it can be easily edited using any of the following ways.

- [Update using editor window](#)
- [Update using saveEvent method](#)

### Update using editor window

You can open the default editor window filled with appointment details by double clicking on the required events. It gets pre-filled with event options such as Subject, Location, Start and End time, All-day, timezone, description and other recurrence options, from which you can edit the desired field values and, then enter the **Save** button to update it.

**Note:** You can also single click on appointments, which opens the quick info popup with edit and delete options. Clicking on the **edit** option will open the default editor filled with event details and **delete** option will prompt for delete confirmation.

### Update using saveEvent method

The appointments can be edited and updated manually using the **saveEvent** method. The following code examples shows how to edit the normal and recurring events.

**Normal event** - Here, an event with ID **3** is edited and its subject is changed with a new text. When the modified data object is passed onto the **saveEvent** method, the changes gets reflected onto the original event. The **Id** field is mandatory in this edit process, where the modified event object should hold the valid **Id** value that exists in the Scheduler data source.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@using Syncfusion.EJ2.Buttons
@(Html.EJS().Button("btn1")
 .Content("EDIT")
 .Render()
)
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var data = {
 Id: 3,
 Subject: 'Testing-edited',
 StartTime: new Date(2018, 1, 11, 10, 0),
 EndTime: new Date(2018, 1, 11, 11, 0),
 IsAllDay: false
 };
 };
```

```

 scheduleObj.saveEvent(data);
 };
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 3,
 Subject = "Testing",
 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
 });
 appData.Add(new AppointmentData
 {
 Id = 4,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 13, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 13, 10, 0, 0),
 IsAllDay = false,
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

**Recurring event** - The following code example shows how to edit a single occurrence of a recurring event. In this case, the modified data should hold an additional field namely **RecurrenceID** mapping to its parent recurring event's Id value. Also, this modified occurrence will be considered as a new event in the Scheduler dataSource, where it is linked with its parent event through the **RecurrenceID** field value.

The `saveEvent` method takes 2 arguments, first one accepting the modified event data object and second argument accepting either of the 2 text values - `EditOccurrence` or `EditSeries`.

When the second argument is passed as `EditOccurrence`, which means that the passed event data is a single modified occurrence - whereas if the second argument is passed as `EditSeries`, it means that the modified data needs to be edited as a whole series and therefore no new event object will be maintained in the Scheduler dataSource.

In case of modifying the single occurrence, it is also necessary to update the `RecurrenceException` field of parent event altogether with the occurrence editing. To know more about how to set `RecurrenceException` values, refer the [recurring events](#) topic.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@using Syncfusion.EJ2.Buttons
@(Html.EJS().Button("btn1")
 .Content("EDIT")
 .Render()
)
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var data = new
ej.data.DataManager(scheduleObj.getCurrentViewEvents()).executeLocal(new
ej.data.Query().where('RecurrenceID', 'equal', 3))[0];
 data.Subject = "Edited";
 scheduleObj.saveEvent(data, 'EditOccurrence');
 };
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
```

```

}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 3,
 Subject = "Testing",
 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3",
 });
 appData.Add(new AppointmentData
 {
 Id = 4,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 12, 11, 0, 0),
 EndTime = new DateTime(2018, 2, 12, 12, 0, 0),
 IsAllDay = false,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=2"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public string RecurrenceRule { get; set; }
}

```

#### Updating events in database at server-side

While editing the normal events in the Scheduler, **update** action takes place and the following code example describes how to update event into database at server side.

```
`sh
```

```
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
```

```
{
```

```
var value = (param.action == "update") ? param.value : param.changed[0];
```

```
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
```

```
if (filterData.Count() > 0)
```

```
{
```

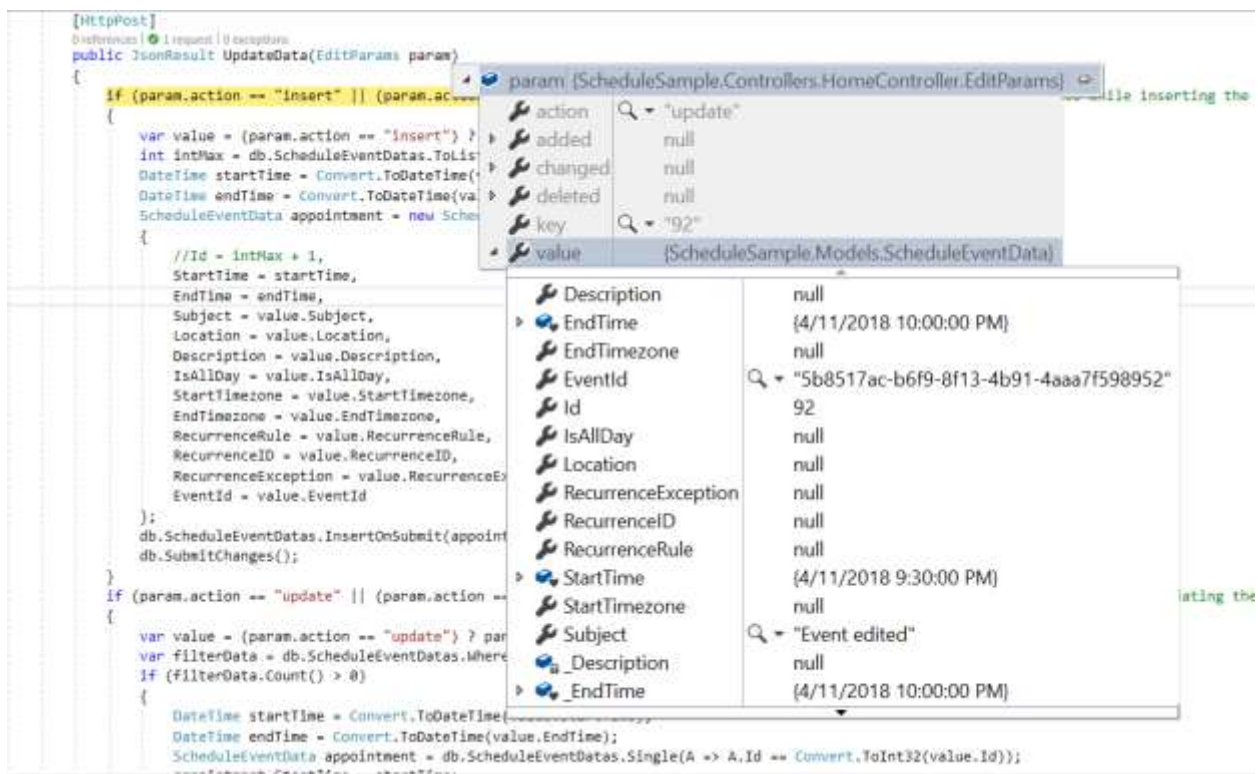
```
DateTime startTime = Convert.ToDateTime(value.StartTime);
```

```
DateTime endTime = Convert.ToDateTime(value.EndTime);
```

```

ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime;
appointment.EndTime = endTime;
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}

```



*How to edit a single occurrence or entire series and update it in database at server-side*  
The recurring appointments can be edited in either of the following two ways.

- Single occurrence

- Entire series

**Editing single occurrence** - When you double click on a recurring event, a popup prompts you to choose either to edit the single event or entire series. From this, if you choose to select **EDIT EVENT** option, a single occurrence of the recurring appointment alone will be edited. The following process takes place while editing a single occurrence,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the **RecurrenceID** field will be added to it, that holds the **Id** value of the parent recurring event. Also, a new **Id** will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate **RecurrenceException** field to hold the edited occurrence appointment's date collection.

Therefore, when a single occurrence is edited from a recurring event, the batch action takes place by allowing both the **Add** and **Edit** action requests to take place together.

**Note:** In case, if you edit an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get updated. In this case, **update** action alone takes place on the edited occurrence object on the database.

```
`sh
```

```
if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
```

```
{
```

```
var value = (param.action == "insert") ? param.value : param.added[0];
```

```
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
1;
```

```
DateTime startTime = Convert.ToDateTime(value.StartTime);
```

```
DateTime endTime = Convert.ToDateTime(value.EndTime);
```

```
ScheduleEventData appointment = new ScheduleEventData()
```

```
{
```

```
Id = intMax + 1,
```

```
StartTime = startTime,
```

```
EndTime = endTime,
```

```
Subject = value.Subject,
```

```
IsAllDay = value.IsAllDay,
```

```
StartTimezone = value.StartTimezone,
```

```
EndTimezone = value.EndTimezone,
```

```
RecurrenceRule = value.RecurrenceRule,
```

```
RecurrenceID = value.RecurrenceID,
```

```

FollowingID = value.FollowingID,
RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime;
appointment.EndTime = endTime;
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.FollowingID = value.FollowingID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}

```

**Editing entire series** - When you select an option **EDIT SERIES** from the popup that opens on double clicking the recurring event, the whole recurring series will be updated with the newly provided value. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences will be removed from the dataSource and simply the single parent data will be updated.



This action of editing entire series also leads to the batch process, as both the **Delete** and **Edit** action takes place together.

```
`sh
```

```
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of code will execute while updating the appointment
```

```
{
```

```
var value = (param.action == "update") ? param.value : param.changed[0];
```

```
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
```

```
if (filterData.Count() > 0)
```

```
{
```

```
DateTime startTime = Convert.ToDateTime(value.StartTime);
```

```
DateTime endTime = Convert.ToDateTime(value.EndTime);
```

```
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id == Convert.ToInt32(value.Id));
```

```
appointment.StartTime = startTime;
```

```
appointment.EndTime = endTime;
```

```
appointment.StartTimezone = value.StartTimezone;
```

```
appointment.EndTimezone = value.EndTimezone;
```

```
appointment.Subject = value.Subject;
```

```
appointment.IsAllDay = value.IsAllDay;
```

```
appointment.RecurrenceRule = value.RecurrenceRule;
```

```
appointment.RecurrenceID = value.RecurrenceID;
```

```
appointment.RecurrenceException = value.RecurrenceException;
```

```
}
```

```
db.SubmitChanges();
```

```
}
```

```
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of code will execute while removing the appointment
```

```
{
```

```
if (param.action == "remove")
```

```
{
```

```
int key = Convert.ToInt32(param.key);
```

```
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
```

```
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
```

```

}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
`

```

**Note:** To know more about handling recurrence exceptions, refer the [Adding exceptions](#) topic.

#### *How to edit from the current and following events of a series*

The recurring appointments can be edited from current and following events when enable the property `editFollowingEvents`.

**Editing Following Events** - When you double click on a recurring event, a popup prompts you to choose either to edit the single event or Edit Following Events or entire series. From this, if you choose to select **EDIT FOLLOWING EVENTS** option, a current and following events of the recurring appointment will be edited. The following process takes place while editing a following events,

- A new event will be created from the parent event data and added to the Scheduler dataSource, with all its default field values overwritten with the newly modified data and additionally, the `followingID` field will be added to it, that holds the `id` value of the immediate parent recurring event. Also, a new `id` will be generated for this event in the dataSource.
- The parent recurring event needs to be updated with appropriate `recurrenceRule` field to hold the modified occurrence appointment's end date.

Therefore, when a following events are edited from a recurring event, the batch action takes place by allowing the `Add`, `Edit` and `Delete` action requests to take place together.

```
`sh
```

```

if (param.action == "insert" || (param.action == "batch" && param.added != null)) // this block of code
will execute while inserting the appointments
{
var value = (param.action == "insert") ? param.value : param.added[0];
int intMax = db.ScheduleEventDatas.ToList().Count > 0 ? db.ScheduleEventDatas.ToList().Max(p => p.Id) :
1;
DateTime startTime = Convert.ToDateTime(value.StartTime);

```

```
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = new ScheduleEventData()
{
 Id = intMax + 1,
 StartTime = startTime,
 EndTime = endTime,
 Subject = value.Subject,
 IsAllDay = value.IsAllDay,
 StartTimezone = value.StartTimezone,
 EndTimezone = value.EndTimezone,
 RecurrenceRule = value.RecurrenceRule,
 RecurrenceID = value.RecurrenceID,
 FollowingID = value.FollowingID,
 RecurrenceException = value.RecurrenceException
};
db.ScheduleEventDatas.InsertOnSubmit(appointment);
db.SubmitChanges();
}

if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
 var value = (param.action == "update") ? param.value : param.changed[0];
 var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
 if (filterData.Count() > 0)
 {
 DateTime startTime = Convert.ToDateTime(value.StartTime);
 DateTime endTime = Convert.ToDateTime(value.EndTime);
 ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
 Convert.ToInt32(value.Id));
 appointment.StartTime = startTime;
 appointment.EndTime = endTime;
 appointment.StartTimezone = value.StartTimezone;
 appointment.EndTimezone = value.EndTimezone;
 appointment.Subject = value.Subject;
 }
}
```

```

appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.FollowingID = value.FollowingID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDats.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDats.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDats.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDats.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
,

```

#### *Restricting edit action based on specific criteria*

You can also dynamically prevent the editing of appointments on Scheduler. For example, say if you want to decline the updating of appointments on non-working hours, you can check for its appropriate condition within the `ActionBegin` event.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")

```

```

.Width("100%")
.Height("550px")
.Views(ViewBag.view)
.ActionBegin("onActionBegin")
.EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
.SelectedDate(new DateTime(2018, 2, 15))
.Render()
)
<script type="text/javascript">
 function onActionBegin(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 if (args.requestType == 'eventChange') {
 var weekEnds = [0, 6];
 var data = args.data;
 var weekDay = weekEnds.indexOf(data.StartTime.getDay()) >= 0;
 var workHours =
((parseInt(scheduleObj.workHours.start.toString().slice(0, 2), 10) <=
data.StartTime.getHours())
 && (parseInt(scheduleObj.workHours.end.toString().slice(0,
2), 10)) >= data.StartTime.getHours());
 if (weekDay || !workHours) {
 args.cancel = true;
 }
 }
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Meteo Showers in 2018", StartTime = new
DateTime(2018, 2, 14, 13, 0, 0), EndTime = new DateTime(2018, 2, 14, 14, 30,
0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 12, 0, 0), EndTime = new DateTime(2018, 2, 15, 14, 0,
0) });
}

```

```

 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

## Delete

The appointments can be deleted in either of the following ways,

- Selecting an appointment and clicking the delete icon from the quick popup that opens.
- Selecting an appointment and pressing **Delete** key.
- Selecting multiple appointments by tap holding an event and then continuously single clicking on other consecutive events and then clicking the **Delete** key.
- Double clicking on an event which opens the default event editor pre-filled with event details, and then choosing **Delete** button in it.

While performing all these above mentioned actions, a pop-up with a delete confirmation message will be displayed prompting either to proceed with deleting an appointment.

### *Deletion using editor window*

When you double click an event, the default editor window will be opened which includes a **Delete** button at the bottom left position to allow you to delete that particular appointment. When deleting an appointment through this editor window, the delete alert confirmation will not be asked and the event will be deleted immediately.

### *Deletion using deleteEvent method*

The appointments can be removed manually using the **deleteEvent** method. The following code examples shows how to edit the normal and recurring events.

**Normal event** - You can delete the normal appointments of Scheduler by simply passing its **Id** value or the entire event object collection to the **deleteEvent** method.

## CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Button("btn1")
 .Content("DELETE")
 .Render()
)
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

```
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.deleteEvent(2);
 };
</script>
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Testing",
 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 13, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 13, 10, 0, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}
```

**Recurring Event** - The recurring events can be removed as an entire series or simply removing single occurrence by using the `deleteEvent` method which takes in either the `DeleteSeries` or `DeleteOccurrence` parameters. The following code example shows how to delete entire series.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Button("btn1")
 .Content("DELETE")
 .Render()
)
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var Data = {
 Id: 2,
 Subject: 'Vacation',
 RecurrenceID: 2,
 StartTime: new Date(2018, 1, 12, 11, 0),
 EndTime: new Date(2018, 1, 12, 12, 0),
 IsAllDay: false,
 RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
 };
 scheduleObj.deleteEvent(Data, 'DeleteSeries');
 };
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
```



```

{
 Id = 1,
 Subject = "Testing",
 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3",
});
appData.Add(new AppointmentData
{
 Id = 2,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 12, 11, 0, 0),
 EndTime = new DateTime(2018, 2, 12, 12, 0, 0),
 IsAllDay = false,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=2"
});
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public string RecurrenceRule { get; set; }
}

```

#### *Removing events from database at server-side*

While deleting the event from the Scheduler, **remove** action takes place and the following code example describes how to delete event from database at server side.

```
`sh
```

```
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
```

```
{
```

```
if (param.action == "remove")
```

```
{
```

```
int key = Convert.ToInt32(param.key);
```

```
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
```

```
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
```

```
}
```

```
else
```

```
{
```

```
foreach (var apps in param.deleted)
```

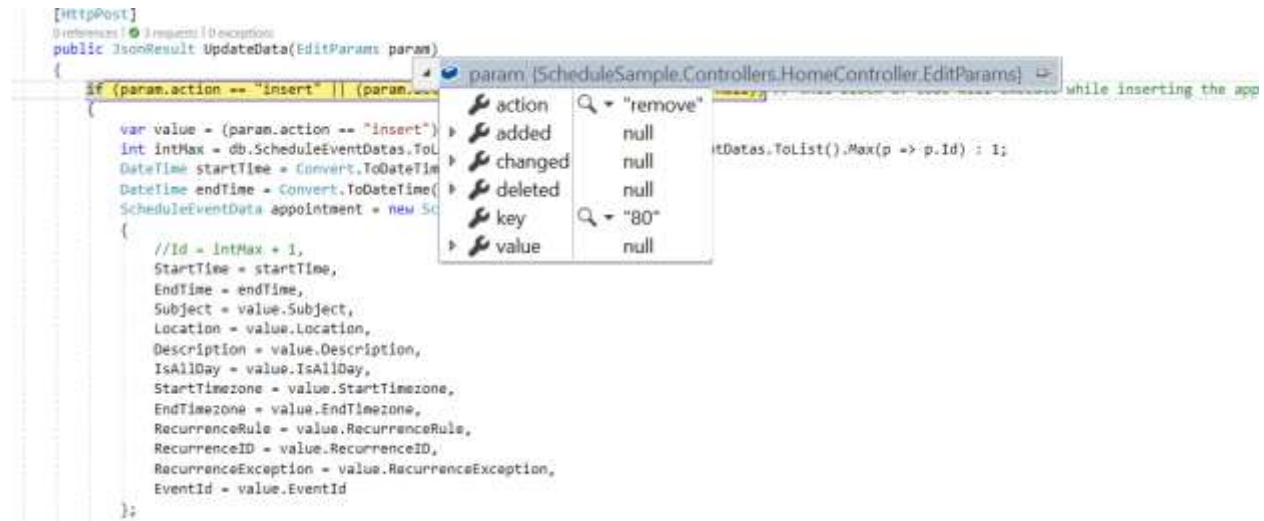
```
{
```

```

ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}

db.SubmitChanges();
}

```



*How to delete a single occurrence or entire series from Scheduler and update it in database at server-side*  
 The recurring events can be deleted in either of the following two ways.

- Single occurrence
- Entire series

**Single occurrence** - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or entire series. From this, if you choose to select **DELETE EVENT** option, a single occurrence of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate **RecurrenceException** field, to hold the deleted occurrence appointment's date collection.

Therefore, when a single occurrence is deleted from a recurring event, the **update** action takes place on the parent recurring event as shown in the following code example.

**Note:** In case, if you delete an existing edited occurrence of a recurring event, only those edited occurrence which present in the database as an individual event object will get removed. In this case, **delete** action takes place instead of **update** action and the parent recurring event object remains same with no changes.

```
`sh
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));
if (filterData.Count() > 0)
{
DateTime startTime = Convert.ToDateTime(value.StartTime);
DateTime endTime = Convert.ToDateTime(value.EndTime);
ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
Convert.ToInt32(value.Id));
appointment.StartTime = startTime;
appointment.EndTime = endTime;
appointment.StartTimezone = value.StartTimezone;
appointment.EndTimezone = value.EndTimezone;
appointment.Subject = value.Subject;
appointment.IsAllDay = value.IsAllDay;
appointment.RecurrenceRule = value.RecurrenceRule;
appointment.RecurrenceID = value.RecurrenceID;
appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}
`
```

**Entire series** - When you select an option **DELETE SERIES** from the popup, the whole recurring series will be deleted. When this option is chosen explicitly, if a parent event holds any edited occurrences - then all its child occurrences which are maintained as separate event objects will also be removed from the dataSource. This action of deleting entire series leads to **remove** action and removes one or more event objects at the same time.

```
`sh
if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
if (param.action == "remove")
```

```

{
int key = Convert.ToInt32(param.key);
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
else
{
foreach (var apps in param.deleted)
{
ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
}
}
db.SubmitChanges();
}
,

```

#### *How to delete only the current and following events of a series*

The recurring events can be deleted from current and following events only when enable `editFollowingEvents` property.

**Delete Following Events** - When you attempt to delete the recurring events, a popup prompts you to choose either to delete the single event or Following Events or entire series. From this, if you choose to select **FOLLOWING EVENT** option, a current and following events of the recurring appointment alone will be removed. The following process takes place while removing a single occurrence,

- The selected occurrence and the following events in same series will be deleted from the Scheduler user interface.
- In code, the parent recurring event object will be updated with appropriate `recurrenceRule` field, to update the end date of the recurring events.

Therefore, when following events are deleted from a recurring event, the `remove` and `update` action takes place on the immediate parent recurring event as shown in the following code example.

```

`sh
if (param.action == "update" || (param.action == "batch" && param.changed != null)) // this block of
code will execute while updating the appointment
{
var value = (param.action == "update") ? param.value : param.changed[0];
var filterData = db.ScheduleEventDatas.Where(c => c.Id == Convert.ToInt32(value.Id));

```

```
if (filterData.Count() > 0)
{
 DateTime startTime = Convert.ToDateTime(value.StartTime);
 DateTime endTime = Convert.ToDateTime(value.EndTime);
 ScheduleEventData appointment = db.ScheduleEventDatas.Single(A => A.Id ==
 Convert.ToInt32(value.Id));
 appointment.StartTime = startTime;
 appointment.EndTime = endTime;
 appointment.StartTimezone = value.StartTimezone;
 appointment.EndTimezone = value.EndTimezone;
 appointment.Subject = value.Subject;
 appointment.IsAllDay = value.IsAllDay;
 appointment.RecurrenceRule = value.RecurrenceRule;
 appointment.RecurrenceID = value.RecurrenceID;
 appointment.FollowingID = value.FollowingID;
 appointment.RecurrenceException = value.RecurrenceException;
}
db.SubmitChanges();
}

if (param.action == "remove" || (param.action == "batch" && param.deleted != null)) // this block of
code will execute while removing the appointment
{
 if (param.action == "remove")
 {
 int key = Convert.ToInt32(param.key);
 ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == key).FirstOrDefault();
 if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
 }
 else
 {
 foreach (var apps in param.deleted)
 {
 ScheduleEventData appointment = db.ScheduleEventDatas.Where(c => c.Id == apps.Id).FirstOrDefault();
 if (appointment != null) db.ScheduleEventDatas.DeleteOnSubmit(appointment);
 }
 }
}
```

```

}
}
db.SubmitChanges();
}
`

```

### Drag and drop

When you drag and drop a normal event on the Scheduler, the event editing action takes place. When a recurring event is drag and dropped on a desired time range, the batch action explained in [Editing a single occurrence](#) process will takes place - thus allowing both the [Add](#) and [Edit](#) action to take place together.

**Note:** By default, when you drag a recurring instance, only the occurrence of the event gets edited and not a whole series.

### Resize

When you resize a normal event on the Scheduler, the event editing action takes place. When a recurring event is resized to a new desired time, the batch action explained in [Editing a single occurrence](#) process will takes place - thus allowing both the [Add](#) and [Edit](#) action to take place together.

**Note:** By default, when you resize a recurring instance, only the occurrence of the event gets edited and not a whole series.

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to knows how to present and manipulate data.

### Virtual scrolling in ASP.NET MVC Schedule control

To achieve better performance in the Scheduler when loading a large number of resources and events, we have added virtual scrolling support to load a large set of resources and events instantly as you scroll. You can dynamically load large number of resources and events in the Scheduler by setting [true](#) to the [allowVirtualScrolling](#) property within the view specific settings. The virtual loading of events is possible in Agenda view, by setting [allowVirtualScrolling](#) property to [true](#) within the agenda view specific settings.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .SelectedDate(new DateTime(DateTime.Today.Year, 4, 1))
 .CurrentView(View.TimelineMonth)
 .Views(view => {
 view.Option(View.TimelineMonth).AllowVirtualScrolling(true).Add();

 view.Option(View.TimelineYear).Orientation(Orientation.Vertical).AllowVirtualScrolling(true).Add();
 })
 .Group(group =>
 group.EnableCompactView(false).Resources(ViewBag.Resource)
 .Resources(res => {

```

```

res.DataSource(ViewBag.resourcesData).Field("ResourceId").Title("Resource").
Name("Resources").TextField("Text").IdField("Id").ColorField("Color").AllowMultiple(true).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = this.generateStaticEvents(new DateTime(2018, 4, 1),
300, 12);
 ViewBag.resourcesData = this.GenerateResourceData(1, 300);
 string[] resources = new string[] { "Resources" };
 ViewBag.Resource = resources;
 return View();
}

private List<ResourceData> GenerateResourceData(int start, int end)
{
 List<ResourceData> resources = new List<ResourceData>(300);
 var colors = new string[] { "#ff8787", "#9775fa", "#748ffc", "#3bc9db",
"#69db7c",
"#fdd835", "#748ffc", "#9775fa", "#df5286", "#7fa900",
"#fec200", "#5978ee", "#00bdae", "#ea80fc" };
 for (int a = start; a <= end; a++)
 {
 int index = a % colors.Length;
 index = index == 0 ? (colors.Length / a):index;
 resources.Add(new ResourceData
 {
 Id = a,
 Text = "Resource " + a,
 Color = colors[index]
 });
 }
 return resources;
}

private List<EventData> generateStaticEvents(DateTime date, int v1, int v2)
{
 List<EventData> data = new List<EventData>(3600);
 var id = 1;
 for (var i = 0; i < 300; i++)
 {
 Random random= new Random();
 List<int> listNumbers = new List<int>();
 int[] randomCollection = new int[24];
 int number;
 int max = 30;
 for (int a = 0; a < 12; a++)
 {
 do
 {

```

```

 number = random.Next(max);
 } while (listNumbers.Contains(number));
 listNumbers.Add(number);
 var startDate = date.AddDays(number);
 startDate = startDate.AddMilliseconds((((number % 10) * 10) *
(1000 * 60)));
 var endDate = startDate.AddMilliseconds(((1440 + 30) * (1000 *
60)));
 data.Add(new EventData
 {
 Id = id,
 Subject = "Event #" + id,
 StartTime = startDate,
 EndTime = endDate,
 IsAllDay = (id % 10 == 0) ? false : true,
 ResourceId = i + 1
 });
 id++;
}
}
return data;
}
class EventData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int ResourceId { get; set; }
}
class ResourceData
{
 public int Id { get; set; }
 public string Text { get; set; }
 public string Color { get; set; }
}

```

**Note:** For now, the virtual loading of resources and events is not supported in **MonthAgenda**, **Year** and **TimelineYear** (Horizontal Orientation) views.

#### Enabling lazy loading for appointments

The lazy loading feature provides a convenient way to efficiently load resource appointments into the Scheduler using an on-demand approach. With this feature, you can seamlessly load a large volume of appointment data into the Scheduler without experiencing any performance degradation.

By default, the Scheduler fetches all the relevant appointments from the server with in the current date range. However, enabling this feature will trigger query requests to the server for appointment retrieval whenever new resources are rendered due to scroll actions. These queries contain the resource IDs of currently displayed resources along with current date range, which can be passed as a comma-separated string. In the server controller, these resource IDs are parsed to filter the necessary appointments to render in the scheduler.



When you enable this feature, the Scheduler becomes capable of fetching events from remote services only for the current view port alone to optimize the data retrieval. The remaining appointment data is fetched from the server on-demand based on currently rendered view port resources as you scroll through the scheduler content.

To enable this feature, you have to set the [EnableLazyLoading](#) property to `true` within the view specific settings.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .SelectedDate(new DateTime(DateTime.Today.Year, 4, 1))
 .CurrentView(View.TimelineMonth)
 .Views(view => {
 view.Option(View.TimelineMonth).EnableLazyLoading(true).Add();
 })
 .Group(group =>
group.EnableCompactView(false).Resources(ViewBag.Resource)
 .Resources(res => {

res.DataSource(ViewBag.resourcesData).Field("ResourceId").Title("Resource").
Name("Resources").TextField("Text").IdField("Id").ColorField("Color").Add();
 }).EventSettings(es => es.DataSource(dataManager =>
 {

dataManager.Url("https://services.syncfusion.com/aspnet/production/api/VirtualEventData").CrossDomain(true).Adaptor("WebApiAdaptor");
 })).Readonly(true)
 .Render()
)
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.resourcesData = this.GenerateResourceData(1, 1000);
 string[] resources = new string[] { "Resources" };
 ViewBag.Resource = resources;
 return View();
}
private List<ResourceData> GenerateResourceData(int start, int end)
{
 List<ResourceData> resources = new List<ResourceData>(300);
 var colors = new string[] { "#ff8787", "#9775fa", "#748ffc", "#3bc9db",
"#69db7c",
"#fdd835", "#748ffc", "#9775fa", "#df5286", "#7fa900",
"#fec200", "#5978ee", "#00bdae", "#ea80fc" };
 for (int a = start; a <= end; a++)
 {
 int index = a % colors.Length;
 index = index == 0 ? (colors.Length / a):index;
 resources.Add(new ResourceData
 {

```

```

 Id = a,
 Text = "Resource " + a,
 Color = colors[index]
 });
}
return resources;
}
class ResourceData
{
 public int Id { get; set; }
 public string Text { get; set; }
 public string Color { get; set; }
}

```

Here's the server-side controller code that retrieves appointment data based on the resource IDs provided as query parameters:

```

`c#
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;
using System;
using Microsoft.EntityFrameworkCore;
using System.Linq;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.OData.Query;
namespace LazyLoadingServices.Controllers
{
 public class VirtualEventDataController : Controller
 {
 private readonly EventsContext dbContext;
 [HttpGet]
 [EnableQuery]
 [Route("api/VirtualEventData")]
 public IActionResult GetData([FromQuery] Params param)
 {
 IQueryable<EventData> query = dbContext.Events;
 // Filter the appointment data based on the ResourceId query params.
 if (!string.IsNullOrEmpty(param.ResourceId))
 {
 string[] resourceId = param.ResourceId.Split(',');
 }

```

```

query = query.Where(data => resourceId.Contains(data.ResourceId.ToString()));
}
return Ok(query.ToList());
}
}
public class Params
{
public DateTime? StartDate { get; set; }
public DateTime? EndDate { get; set; }
public string ResourceId { get; set; }
}
}
`

```

**Note:**

- The property will be effective, when large number of resources and appointments bound to the Scheduler.
- This property is applicable only when [resource grouping](#) is enabled in Scheduler.

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Editor window and quick popups

Scheduler makes use of popups and dialog to display the required notifications, as well as includes an editor window with event fields for making the appointment creation and editing process easier. You can also easily customize the editor window and the fields present in it, and can also apply validations on those fields.

#### Event editor

The editor window usually opens on the Scheduler, when a cell or event is double clicked. When a cell is double clicked, the detailed editor window opens in "Add new" mode, whereas when an event is double clicked, the same is opened in an "Edit" mode.

In mobile devices, you can open the detailed editor window in edit mode by clicking the edit icon on the popup, that opens on single tapping an event. You can also open it in add mode by single tapping a cell, which will display a **+** indication, clicking on it again will open the editor window.

**Note:** You can also prevent the editor window from opening, by rendering Scheduler in a **Readonly** mode or by doing code customization within the **PopupOpen** event.

#### *How to change the editor window header title and text of footer buttons*

You can change the header title and the text of buttons displayed at the footer of the editor window by changing the appropriate localized word collection used in the Scheduler.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 var L10n = ej.base.L10n;
 L10n.load({
 'en-US': {
 'schedule': {
 'saveButton': 'Add',
 'cancelButton': 'Close',
 'deleteButton': 'Remove',
 'newEvent': 'Add Event',
 },
 },
 });
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
}

```

```

public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
}

```

#### *How to change the label text of default editor fields*

To change the default labels such as Subject, Location and other field names in the editor window, make use of the **Title** property available within the field option of **EventSettings**.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .SelectedDate(new DateTime(2018, 2, 15))
 .EventSettings(e => e.Fields(f => f.Subject(sub =>
sub.Name("Subject").Title("Summary"))
 .Location(loc => loc.Name("Location").Title("Location"))
 .Description(des => des.Name("Description").Title("Comments"))
 .StartTime(st => st.Name("StartTime").Title("From"))
 .EndTime(et => et.Name("EndTime").Title("To"))
)
 .DataSource(ViewBag.datasource)
)
.Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}

```

```

}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

### Field validation

It is possible to validate the required fields of the editor window from client-side before submitting it, by adding appropriate validation rules to each field. The appointment fields have been extended to accept both **string** and **object** type values. To perform validations, it is necessary to specify object values for the event fields.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@{
 var validationRules = new Dictionary<string, object>() { { "required",
true } };
 var locationValidationRules = new Dictionary<string, object>() { {
"required", true }, { "regex", new string[] { "[a-zA-Z0-9-]*$", "Special
character(s) not allowed in this field" } } };
 var descriptionValidationRules = new Dictionary<string, object>() { {
"required", true }, { "minLength", 5 }, { "maxLength", 500 } };
}
@(Html.EJS().Schedule("schedule")
.Width("100%")
.Height("650px")
.EventSettings(e => e.Fields(f => f.Subject(sub =>
sub.Name("Subject").Validation(validationRules))
.Location(loc =>
loc.Name("Location").Validation(locationValidationRules))
.Description(des =>
des.Name("Description").Validation(descriptionValidationRules))
.StartTime(st =>
st.Name("StartTime").Validation(validationRules))
.EndTime(et => et.Name("EndTime").Validation(validationRules))
)
.DataSource(ViewBag.datasource)
)
.SelectedDate(new DateTime(2018, 2, 15))
.Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 }
}

```

```

 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

**Note:** Applicable validation rules can be referred from [form validation](#) documentation.

#### Add additional fields to the default editor

The additional fields can be added to the default event editor by making use of the **PopupOpen** event which gets triggered before the event editor opens on the Scheduler. The **PopupOpen** is a client-side event that triggers before any of the generic popups opens on the Scheduler. The additional field (any of the form elements) should be added with a common class name **e-field**, so as to handle and process those additional data along with the default event object. In the following example, an additional field **Event Type** has been added to the default event editor and its value is processed accordingly.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 // Create required custom elements in initial time

```

```

 if (!args.element.querySelector('.custom-field-row')) {
 var row = ej.base.createElement('div', { className: 'custom-
field-row' });
 var formElement = args.element.querySelector('.e-schedule-
form');
 formElement.firstChild.insertBefore(row,
args.element.querySelector('.e-title-location-row'));
 var container = ej.base.createElement('div', { className:
'custom-field-container' });
 var inputEle = ej.base.createElement('input', {
 className: 'e-field', attrs: { name: 'EventType' }
 });
 container.appendChild(inputEle);
 row.appendChild(container);
 var dropDownList = new ej.dropdowns.DropDownList({
 dataSource: [
 { text: 'Public Event', value: 'public-event' },
 { text: 'Maintenance', value: 'maintenance' },
 { text: 'Commercial Event', value: 'commercial-
event' },
 { text: 'Family Event', value: 'family-event' }
],
 fields: { text: 'text', value: 'value' },
 value: (args.data).EventType,
 floatLabelType: 'Always', placeholder: 'Event Type'
 });
 dropDownList.appendTo(inputEle);
 inputEle.setAttribute('name', 'EventType');
 }
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 });
}

```



```

 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

#### *Customizing the default time duration in editor window*

In default event editor window, start and end time duration are processed based on the **interval** value set within the **TimeScale** property. By default, **interval** value is set to 30, and therefore the start/end time duration within the event editor will be in a 30 minutes time difference. You can change this duration value by changing the **duration** option within the **PopupOpen** event as shown in the following code example.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 args.duration = 60;
 }
 }
</script>

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

```

```

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

#### *How to prevent the display of editor and quick popups*

It is possible to prevent the display of editor and quick popup windows by passing the value `true` to `cancel` option within the `PopupOpen` event.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onPopupOpen(args) {
 args.cancel = true;
 }
</script>

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 }
}

```

```

 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the **PopupOpen** event are as follows.

| Type            | Description                                     |
|-----------------|-------------------------------------------------|
| Editor          | For Detailed editor window.                     |
| QuickInfo       | For Quick popup which opens on cell click.      |
| EditEventInfo   | For Quick popup which opens on event click.     |
| ViewEventInfo   | For Quick popup which opens on responsive mode. |
| EditorContainer | For more event indicator popup.                 |

### Quick popups

The quick info popups are the ones that gets opened, when a cell or appointment is single clicked on the desktop mode. On single clicking a cell, you can simply provide a subject and save it. Also, while single clicking on an event, a popup will be displayed where you can get the overview of the event information. You can also edit or delete those events through the options available in it.

By default, these popups are displayed over cells and appointments of Scheduler and to disable this action, set **false** to **ShowQuickInfo** property.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")

```

```

 .Width("100%")
 .Height("550px")
 .ShowQuickInfo(false)
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

**Note:** The quick popup that opens while single clicking on the cells are not applicable on mobile devices.

### [How to open Quick Info popup on multiple cell selection](#)

By default the **QuickInfo** popup will open on single click of the cell. To open the quick info popup on multiple cell selection, you need to select the cells and press **enter** key. You can open this popup immediately after multiple cell selection by setting up **true** to **QuickInfoOnSelectionEnd** property where as its default value is **false**.

## CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS() .Schedule ("schedule")
 .Width ("100%")
 .Height ("550px")
 .QuickInfoOnSelectionEnd (true)
 .Render ()
)
```

### DATA.CS

```
public ActionResult Index()
{
 return View();
}
```

#### *How to change the watermark text of quick popup subject*

By default, **Add Title** text is displayed on the subject field of quick popup. To change the default watermark text, change the value of the appropriate localized word collection used in the Scheduler.

```
`csharp
var L10n = ej.base.L10n;
L10n.load({
 'en-US': {
 'schedule': {
 'addTitle' : 'New Title'
 }
 }
});
`
```

#### *Customizing quick popups*

The look and feel of the built-in quick popup window, which opens when single clicked on the cells or appointments can be customized by making use of the **QuickInfoTemplates** property of the Scheduler. There are 3 sub-options available to customize them easily,

- header - Accepts the template design that customizes the header part of the quick popup.
- content - Accepts the template design that customizes the content part of the quick popup.
- footer - Accepts the template design that customizes the footer part of the quick popup.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS() .Schedule ("schedule")
 .Width ("100%")
 .Height ("550px")
 .QuickInfoTemplates (qt => qt.Header ("#headerTemplate")
 .Content ("#contentTemplate")
)
)
```

```

 .Footer("#footerTemplate")
)
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .e-textlabel {
 font-weight: bold;
 padding-right: 5px;
 }
 .custom-event-editor td {
 padding: 5px;
 }
 .e-quick-popup-wrapper .e-cell-content {
 padding: 0 10px 10px 10px;
 }
 .e-quick-popup-wrapper .e-cell-content div {
 padding-bottom: 10px;
 }
 .e-quick-popup-wrapper .e-cell-content .e-field {
 border-left-width: 0;
 border-top-width: 0;
 border-right-width: 0;
 height: 25px;
 width: 100%;
 }
 .e-quick-popup-wrapper .e-event-content {
 display: flex;
 }
 .e-quick-popup-wrapper .e-event-content img {
 width: 100px;
 }
 .e-quick-popup-wrapper .e-event-header {
 position: absolute;
 right: 0;
 }
 .e-quick-popup-wrapper .e-cell-footer .e-event-create,
 .e-quick-popup-wrapper .e-event-footer .e-event-edit {
 position: absolute;
 right: 0;
 }
 .e-quick-popup-wrapper .e-event-footer .e-event-delete {
 padding-left: 100px;
 }
 .e-quick-popup-wrapper .e-event-content .subject {
 padding-top: 30px;
 font-weight: 500;
 font-size: 14px;
 }
</style>
<script id="headerTemplate" type="text/x-template">
 <div>
 ${if (elementType === 'cell')}
 <div class="e-cell-header">

```

```

 <div class="e-header-icon-wrapper">
 <button class="e-close" title="Close"></button>
 </div>
 </div>
 ${else}
 <div class="e-event-header">
 <div class="e-header-icon-wrapper">
 <button class="e-close" title="CLOSE"></button>
 </div>
 </div>
 ${/if}
</div>
</script>
<script id="contentTemplate" type="text/x-template">
 <div>
 ${if (elementType === 'cell')}
 <div class="e-cell-content">
 <form class="e-schedule-form">
 <div>
 <input class="subject e-field" type="text"
name="Subject" placeholder="Title">
 </div>
 <div>
 <input class="location e-field" type="text"
name="Location" placeholder="Location">
 </div>
 </form>
 </div>
 ${else}
 <div class="e-event-content">
 <div class="e-subject-wrap">
 ${if (Subject)}
 <div class="subject">${Subject}</div>${/if} ${if (Location)}
 <div class="location">${Location}</div>${/if} ${if
(Description)}
 <div class="description">${Description}</div>${/if}
 </div>
 </div>
 ${/if}
 </div>
</script>
<script id="footerTemplate" type="text/x-template">
 <div>
 ${if (elementType === 'cell')}
 <div class="e-cell-footer">
 <button class="e-event-details" title="Extra Details">Extra
Details</button>
 <button class="e-event-create" title="Add">Add</button>
 </div>
 ${else}
 <div class="e-event-footer">
 <button class="e-event-edit" title="Edit">Edit</button>
 <button class="e-event-delete" title="Delete">Delete</button>
 </div>
 ${/if}
 </div>
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

**Note:** The quick popup in adaptive mode can also be customized using `QuickInfoTemplates` using `e-device` class.

*Customizing timezone collection in the editor window*

By default, the timezone collections in the editor window have been loaded with built-in timezone collections. Now we can be able to customize the timezone collections using the `TimezoneDataSource` property with the collection of `TimezoneFields` data.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .TimezoneDataSource : ([
 { Value: 'Pacific/Niue', Text: 'Niue' },

```



```

 { Value: 'Pacific/Pago_Pago', Text: 'Pago Pago' },
 { Value: 'Pacific/Honolulu', Text: 'Hawaii Time' },
 { Value: 'Pacific/Rarotonga', Text: 'Rarotonga' },
 { Value: 'Pacific/Tahiti', Text: 'Tahiti' },
])
 .SelectedDate(new DateTime(2020, 1, 15))
 .Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Explosion of Betelgeuse Star",
 StartTime= new DateTime(2023, 1, 15, 10, 0, 0),
 EndTime= new DateTime(2023, 1, 15, 12, 30, 0),
 IsAllDay= false
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Blue Moon Eclipse",
 StartTime = new DateTime(2023, 1, 16, 12, 0, 0),
 EndTime = new DateTime(2023, 1, 16, 13, 0, 0),
 IsAllDay = false
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

## Customizing event editor using template

The event editor window can be customized by making use of the `EditorTemplate` option. Here, the custom window design is built with the required fields using the script template and its type should be of **text/x-template**.

Each field defined within template should contain the **e-field** class, so as to allow the processing of those field values internally. The ID of this customized script template section is assigned to the

**EditorTemplate** option, so that these customized fields will be replaced onto the default editor window.

Note: **e-field** class only applicable for **DropDownList**, **DateTimePicker**, **MultiSelect**, **DatePicker**, **CheckBox** and **TextBox** components. Since we have processed the field values internally for the above mentioned components.

As we are using our Syncfusion sub-components within our editor using template in the following example, the custom defined form elements needs to be configured as required Syncfusion components such as **DropDownList** and **DateTimePicker** within the **PopupOpen** event. This particular step can be skipped, if the user needs to simply use the usual form elements.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .EditorTemplate("#EventEditorTemplate")
 .ShowQuickInfo(false)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .custom-event-editor .e-textlabel {
 padding-right: 15px;
 text-align: right;
 }
 .custom-event-editor td {
 padding: 7px;
 padding-right: 16px;
 }
</style>
<script id="EventEditorTemplate" type="text/template">
 <table class="custom-event-editor" width="100%" cellpadding="5">
 <tbody>
 <tr>
 <td class="e-textlabel">Summary</td>
 <td colspan="4">
 <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Status</td>
 <td colspan="4">
 <input type="text" id="EventType" name="EventType"
class="e-field" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">From</td>
```

```

 <td colspan="4">
 <input id="StartTime" class="e-field" type="text"
name="StartTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">To</td>
 <td colspan="4">
 <input id="EndTime" class="e-field" type="text"
name="EndTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Reason</td>
 <td colspan="4">
 <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50" style="width: 100%; height: 60px
!important; resize: vertical"></textarea>
 </td>
 </tr>
</tbody>
</table>
</script>
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 var statusElement = args.element.querySelector('#EventType');
 if (!statusElement.classList.contains('e-dropdownlist')) {
 var dropDownListObject = new ej.dropdowns.DropDownList({
 placeholder: 'Choose status', value:
statusElement.value,
 dataSource: ['New', 'Requested', 'Confirmed']
 });
 dropDownListObject.appendTo(statusElement);
 statusElement.setAttribute('name', 'EventType');
 }
 var startElement = args.element.querySelector('#StartTime');
 if (!startElement.classList.contains('e-datetimepicker')) {
 new ej.calendars.DateTimePicker({ value: new
Date(startElement.value) || new Date() }, startElement);
 }
 var endElement = args.element.querySelector('#EndTime');
 if (!endElement.classList.contains('e-datetimepicker')) {
 new ej.calendars.DateTimePicker({ value: new
Date(endElement.value) || new Date() }, endElement);
 }
 }
 }
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()

```

```

 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

#### *How to customize header and footer using template*

The editor window's header and footer can be enhanced with custom designs using the [EditorHeaderTemplate](#) and [EditorFooterTemplate](#) options. To achieve this, create a script template that includes the necessary fields. Ensure that the template type is set to **text/x-template**.

In this demo, we tailor the editor's header according to the appointment's subject field using the [EditorHeaderTemplate](#). Furthermore, we make use of the [EditorFooterTemplate](#) to handle the functionality of validating specific fields before proceeding with the save action or canceling it if validation requirements are not

met.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("scheduler")
.Width("100%")
.Height("550px")
.PopupOpen("onPopupOpen")
.EventSettings(new ScheduleEventSettings { DataSource = ViewBag.datasource
}))
.EditorHeaderTemplate("#editor-header")
.EditorFooterTemplate("#editor-footer")
.Render()
)

```

```

<style>
#verify {
 position: fixed;
 padding: 0 20px;
}
#text {
 cursor: pointer;
 display: inline-block;
 font-family: "Roboto", -apple-system, BlinkMacSystemFont, "Segoe UI",
"Helvetica Neue", sans-serif;
 font-size: 14px;
 font-weight: normal;
 line-height: 14px;
 user-select: none;
 margin-left: 8px;
 vertical-align: middle;
 white-space: normal;
}
#right-button {
 padding: 0 10px;
}
</style>
<script id="editor-header" type="text/x-template">
 ${if (!Subject)}<div>Create New
Event</div>${else}<div>${Subject}</div>${/if}
</script>
<script id="editor-footer" type="text/x-template">
 <div id="verify">
 <input type="checkbox" id="check-box" value="unchecked">
 <label id="text">Verified</label>
 </div>
 <div id="right-button">
 <button id="Save" class="e-control e-btn e-primary" disabled data-
ripple="true">Save</button>
 <button id="Cancel" class="e-control e-btn e-primary" data-
ripple="true">Cancel</button>
 </div>
</script>
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 let scheduleObj =
document.getElementById('scheduler').ej2_instances[0];
 const saveButton = args.element.querySelector("#Save");
 const cancelButton = args.element.querySelector("#Cancel");
 const checkBox = args.element.querySelector("#check-box");
 checkBox.onChange = () => {
 if (!checkBox.checked) {
 saveButton.setAttribute("disabled", "");
 } else {
 saveButton.removeAttribute("disabled");
 }
 };
 saveButton.onclick = () => {
 const data = {
 Id: args.data.Id,
 Subject: args.element.querySelector("#Subject").value,

```

```

 StartTime:
args.element.querySelector("#StartTime").ej2_instances[0].value,
 EndTime:
args.element.querySelector("#EndTime").ej2_instances[0].value,
 IsAllDay:
args.element.querySelector("#IsAllDay").checked,
 };
 if (args.target.classList.contains("e-appointment")) {
 scheduleObj.saveEvent(data, "Save");
 } else {
 data.Id = scheduleObj.getEventMaxID();
 scheduleObj.addEvent(data);
 }
 scheduleObj.closeEditor();
};
cancelButton.onclick = () => {
 scheduleObj.closeEditor();
};
 }
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Surgery - Andrew",
 StartTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 9, 0, 0),
 EndTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 10, 0, 0),
 IsAllDay = false,
 },
 {
 Id: 2,
 Subject: "Consulting - John",
 StartTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 10, 0, 0),
 EndTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 11, 30, 0),
 IsAllDay: false
 },
 {
 Id: 3,
 Subject: "Therapy - Robert",
 StartTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 11, 30, 0),
 }
);
}

```

```

 EndTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 12, 30, 0),
 IsAllDay: false
 },
 {
 Id: 4,
 Subject: "Observation - Steven",
 StartTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 12, 30, 0),
 EndTime = new DateTime(DateTime.Today.Year, DateTime.Today.Month,
DateTime.Today.Day, 13, 30, 0),
 IsAllDay: false
 }
);
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

!{

#### [How to add resource options within editor template](#)

The resource field can be added within editor template with MultiSelect control for allow multiple resources.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res =>
 {
 res.DataSource(ViewBag.Owners)
 .Field("OwnerId")
 .Title("Owners")
 .Name("Owners")
 .AllowMultiple(true)
 .TextField("text")
 .IdField("id")
 .ColorField("color")
 .Add();
 })
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .EditorTemplate("#EventEditorTemplate")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()

```

```

)
<style>
 .custom-event-editor .e-textlabel {
 padding-right: 15px;
 text-align: right;
 }
 .custom-event-editor td {
 padding: 7px;
 padding-right: 16px;
 }
</style>
<script id="EventEditorTemplate" type="text/x-template">
 <table class="custom-event-editor" width="100%" cellpadding="5">
 <tbody>
 <tr>
 <td class="e-textlabel">Summary</td>
 <td colspan="4">
 <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">From</td>
 <td colspan="4">
 <input id="StartTime" class="e-field" type="text"
name="StartTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">To</td>
 <td colspan="4">
 <input id="EndTime" class="e-field" type="text"
name="EndTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Owner</td>
 <td colspan="4">
 <input type="text" id="OwnerId" name="OwnerId" class="e-
field" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Reason</td>
 <td colspan="4">
 <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50" style="width: 100%; height: 60px
!important; resize: vertical">
 </td>
 </tr>
 </tbody>
 </table>
</script>
<script type="text/javascript">
 function onPopupOpen (args) {
 if (args.type === 'Editor') {

```



```

 var ownerData = @Html.Raw(Json.Encode(ViewBag.Owners));
 var startElement = args.element.querySelector('#StartTime');
 if (!startElement.classList.contains('e-datetimepicker')) {
 new ej.calendars.DateTimePicker({ value: new
Date(startElement.value) || new Date() }, startElement);
 }
 var endElement = args.element.querySelector('#EndTime');
 if (!endElement.classList.contains('e-datetimepicker')) {
 new ej.calendars.DateTimePicker({ value: new
Date(endElement.value) || new Date() }, endElement);
 }
 var processElement = args.element.querySelector('#OwnerId');
 if (!processElement.classList.contains('e-multiselect')) {
 var multiSelectObject = new ej.dropdowns.MultiSelect({
 placeholder: 'Select a owner',
 dataSource: ownerData,
 fields: { text: 'text', value: 'id' },
 value: (args.data.OwnerId instanceof Array) ?
args.data.OwnerId : [args.data.OwnerId]
 });
 multiSelectObject.appendTo(processElement);
 }
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetEventsData();
 List<OwnerRes> owners = new List<OwnerRes>();
 owners.Add(new OwnerRes { text = "Nancy", id = 1, color = "#1aaa55" });
 owners.Add(new OwnerRes { text = "Smith", id = 2, color = "#7fa900" });
 owners.Add(new OwnerRes { text = "Paul", id = 3, color = "#357cd2" });
 ViewBag.Owners = owners;
 string[] resources = new string[] { "Owners" };
 ViewBag.Resources = resources;
 return View();
}

public class OwnerRes
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}

public List<EventsData> GetEventsData()
{
 List<EventsData> eventsData = new List<EventsData>();
 eventsData.Add(new EventsData
 {
 Id = 1,
 Subject = "Surgery - Nancy",
 StartTime = new DateTime(2023, 2, 11, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 11, 12, 0, 0),
 EventType = "Confirmed",
 });
}

```

```

 OwnerId = 1
 });
 eventsData.Add(new EventsData
 {
 Id = 2,
 Subject = "Therapy - Smith",
 StartTime = new DateTime(2023, 2, 12, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 12, 12, 0, 0),
 EventType = "New",
 OwnerId = 2
 });
 eventsData.Add(new EventsData
 {
 Id = 3,
 Subject = "Surgery - Paul",
 StartTime = new DateTime(2023, 2, 13, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 13, 12, 0, 0),
 EventType = "Requested",
 OwnerId = 3
 });
 return eventsData;
}
public class EventsData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string EventType { get; set; }
 public int OwnerId { get; set; }
}

```

#### *Apply validations on editor template fields*

In the following code example, validation has been added to the status field.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .PopupOpen("onPopupOpen")
 .EditorTemplate("#EventEditorTemplate")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .custom-event-editor .e-textlabel {
 padding-right: 15px;
 text-align: right;
 }
 .custom-event-editor td {
 padding: 7px;
 padding-right: 16px;
 }

```

```

 }
</style>
<script id="EventEditorTemplate" type="text/x-template">
 <table class="custom-event-editor" width="100%" cellpadding="5">
 <tbody>
 <tr>
 <td class="e-textlabel">Summary</td>
 <td colspan="4">
 <input id="Subject" class="e-field e-input" type="text"
value="" name="Subject" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Status</td>
 <td colspan="4">
 <input type="text" id="EventType" name="EventType"
class="e-field" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">From</td>
 <td colspan="4">
 <input id="StartTime" class="e-field" type="text"
name="StartTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">To</td>
 <td colspan="4">
 <input id="EndTime" class="e-field" type="text"
name="EndTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Reason</td>
 <td colspan="4">
 <textarea id="Description" class="e-field e-input"
name="Description" rows="3" cols="50" style="width: 100%; height: 60px
!important; resize: vertical"></textarea>
 </td>
 </tr>
 </tbody>
 </table>
</script>
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 if
(!ej.base.isNullOrUndefined(document.getElementById("EventType_Error"))) {
 document.getElementById("EventType_Error").style.display =
"none";
 document.getElementById("EventType_Error").style.left =
"351px";
 }
 var formElement = args.element.querySelector('.e-schedule-
form');
 var statusElement = args.element.querySelector('#EventType');

```

```

 if (!statusElement.classList.contains('e-dropdownlist')) {
 var dropDownListObject = new ej.dropdowns.DropDownList({
 placeholder: 'Select a status', value:
statusElement.value,
 dataSource: ['New', 'Requested', 'Confirmed'],
 select: valueChange
 });
 dropDownListObject.appendTo(statusElement);
 }
 function valueChange() {
 if
(!ej.base.isNullOrUndefined(document.getElementById("EventType_Error"))) {
 document.getElementById("EventType_Error").style.display
= "none";
 }
 }
 var validator = ((formElement).ej2_instances[0]);
 validator.addRules('EventType', { required: true });
 var startElement = args.element.querySelector('#StartTime');
 if (!startElement.classList.contains('e-datetimepicker')) {
 new ej.calendars.DateTimePicker({ value: new
Date(startElement.value) || new Date() }, startElement);
 }
 var endElement = args.element.querySelector('#EndTime');
 if (!endElement.classList.contains('e-datetimepicker')) {
 new ej.calendars.DateTimePicker({ value: new
Date(endElement.value) || new Date() }, endElement);
 }
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
}

```

```

public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
}

```

#### *How to save the customized event editor using template*

The **e-field** class is not added to each field defined within the template, so you should allow to set those field values externally by using the **popupClose** event.

**Note:** You can allow to retrieve the data only on the **save** and **delete** option. Data cannot be retrieved on the **close** and **cancel** options in the editor window.

The following code example shows how to save the customized event editor using a template by the **popupClose** event.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .PopupOpen("onPopupOpen")
 .PopupClose("onPopupClose")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .EditorTemplate("#EventEditorTemplate")
 .ShowQuickInfo(false)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .custom-event-editor .e-textlabel {
 padding-right: 15px;
 text-align: right;
 }
 .custom-event-editor td {
 padding: 7px;
 padding-right: 16px;
 }
</style>
<script id="EventEditorTemplate" type="text/template">
 <table class="custom-event-editor" width="100%" cellpadding="5">
 <tbody>
 <tr>
 <td class="e-textlabel">Summary</td>
 <td colspan="4">
 <input id="Subject" class="e-input" type="text"
name="Subject" style="width: 100%" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Status</td>
 <td colspan="4">
 <input type="text" id="EventType" name="EventType"
style="width: 100%" />
 </td>
 </tr>
 </tbody>
 </table>
</script>

```

```

 </td>
 </tr>
 <tr>
 <td class="e-textlabel">From</td>
 <td colspan="4">
 <input id="StartTime" type="text" name="StartTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">To</td>
 <td colspan="4">
 <input id="EndTime" type="text" name="EndTime" />
 </td>
 </tr>
 <tr>
 <td class="e-textlabel">Reason</td>
 <td colspan="4">
 <textarea id="Description" class="e-input"
name="Description" rows="3" cols="50" style="width: 100%; height: 60px
!important; resize: vertical"></textarea>
 </td>
 </tr>
</tbody>
</table>
</script>
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 var subjectElement = args.element.querySelector('#Subject');
 if (subjectElement) {
 subjectElement.value = (args.data).Subject || "";
 }
 var statusElement =
args.element.querySelector('#EventType');
 if (!statusElement.classList.contains('e-dropdownlist')) {
 var dropDownListObject = new ej.dropdowns.DropDownList({
 placeholder: 'Choose status', value:
(args.data).EventType,
 dataSource: ['New', 'Requested', 'Confirmed']
 });
 dropDownListObject.appendTo(statusElement);
 }
 var startElement = args.element.querySelector('#StartTime');
 if (!startElement.classList.contains('e-datetimepicker')) {
 startElement.value = (args.data).StartTime;
 new ej.calendars.DateTimePicker({ value: new
Date(startElement.value) || new Date() }, startElement);
 }
 var endElement = args.element.querySelector('#EndTime');
 if (!endElement.classList.contains('e-datetimepicker')) {
 endElement.value = (args.data).EndTime;
 new ej.calendars.DateTimePicker({ value: new
Date(endElement.value) || new Date() }, endElement);
 }
 var descriptionElement =
args.element.querySelector('#Description');
 if (descriptionElement) {

```

```

descriptionElement.value = (args.data).Description ||
"";
 }
 }
 }
 function onPopupClose(args) {
 if (args.type === 'Editor' &&
!ej.base.isNullOrUndefined(args.data)) {
 var subjectElement = args.element.querySelector('#Subject');
 if (subjectElement) {
 (args.data).Subject = subjectElement.value;
 }
 var statusElement =
args.element.querySelector('#EventType');
 if (statusElement) {
 (args.data).EventType = statusElement.value;
 }
 var startElement = args.element.querySelector('#StartTime');
 if (startElement) {
 (args.data).StartTime = startElement.value;
 }
 var endElement = args.element.querySelector('#EndTime');
 if (endElement) {
 (args.data).EndTime = endElement.value;
 }
 var descriptionElement =
args.element.querySelector('#Description');
 if (descriptionElement) {
 (args.data).Description = descriptionElement.value;
 }
 }
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Surgery - Andrew",

```

```

 EventType = "Confirmed",
 StartTime = new DateTime(2018, 2, 12, 9, 30, 0),
 EndTime = new DateTime(2018, 2, 12, 10, 30, 0),
 OwnerId = 3
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string EventType { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
}

```

In case, if you need to prevent only specific popups on Scheduler, then you can check the condition based on the popup type. The types of the popup that can be checked within the `popupClose` event are as follows.

| Type                      | Description                                     |
|---------------------------|-------------------------------------------------|
| ----- -----               |                                                 |
| Editor                    | For Detailed editor window.                     |
| QuickInfo                 | For Quick popup which opens on cell click.      |
| EditEventInfo             | For Quick popup which opens on event click.     |
| ViewEventInfo             | For Quick popup which opens on responsive mode. |
| EventContainer            | For more event indicator popup.                 |
| RecurrenceAlert           | For edit recurrence event alert popup.          |
| DeleteAlert               | For delete confirmation popup.                  |
| ValidationAlert           | For validation alert popup.                     |
| RecurrenceValidationAlert | For recurrence validation alert popup.          |

#### More events indicator and popup

When the number of appointments count that lies on a particular time range \* default appointment height exceeds the default height of a cell in month view and all other timeline views, a + more text indicator will be displayed at the bottom of those cells. This indicator denotes that the cell contains few more appointments in it and clicking on that will display a popup displaying all the appointments present on that day.

**Note:** To disable this option of showing popup with all hidden appointments, while clicking on the text indicator, you can do code customization within the `PopupOpen` event.

The same indicator is displayed on all-day row in calendar views such as day, week and work week views alone, when the number of appointment count present in a cell exceeds three. Clicking on the text



indicator here will not open a popup, but will allow the expand/collapse option for viewing the remaining appointments present in the all-day row.

The following code example shows how to disable the display of such popups while clicking on the more text indicator.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .CurrentView(View.Month)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'EventContainer') {
 args.cancel = true;
 }
 }
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData{ Id = 1, Subject = "Meeting-1",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData{ Id = 2, Subject = "Meeting-2",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData{ Id = 3, Subject = "Meeting-3",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
}
```

```

 appData.Add(new AppointmentData{ Id = 4, Subject = "Meeting-4",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 return appData;
 }
 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 }

```

#### *How to customize the popup that opens on more indicator*

The following code example shows you how to customize the default more indicator popup in which number of events rendered count on the day has been shown in the header.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .CurrentView(View.Month)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'EventContainer') {
 var instance = new ej.base.Internationalization();
 var date = instance.formatDate(args.data.date, { skeleton:
'MMEd' });
 args.element.querySelector('.e-header-date').innerText = date;
 args.element.querySelector('.e-header-day').innerText = 'Event
count: ' + args.data.event.length;
 }
 }
</script>

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 }
}

```

```

 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData{ Id = 1, Subject = "Meeting-1",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData{ Id = 2, Subject = "Meeting-2",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData{ Id = 3, Subject = "Meeting-3",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData{ Id = 4, Subject = "Meeting-4",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0), EndTime
 = new DateTime(2023, 2, 15, 12, 30, 0), IsAllDay = true });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

#### *How to prevent the display of popup when clicking on the more text indicator*

It is possible to prevent the display of popup window by passing the value `true` to `cancel` option within the `MoreEventsClick` event.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .CurrentView(View.Month)
 .MoreEventsClick("onMoreEventsClick")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onMoreEventsClick(args) {
 args.cancel = true;
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData { Id = 1, Subject = "Meeting-1",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData { Id = 2, Subject = "Meeting-2",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData { Id = 3, Subject = "Meeting-3",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData { Id = 4, Subject = "Meeting-4",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

*How to navigate Day view when clicking on more text indicator*

The following code example shows you how to customize the `moreEventsClick` property to navigate to the Day view when clicking on the more text indicator.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .CurrentView(View.Month)
 .MoreEventsClick("onMoreEventsClick")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })

```

```

 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
 <script type="text/javascript">
 function onMoreEventsClick(args) {
 args.isPopupOpen = false;
 }
 </script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData { Id = 1, Subject = "Meeting-1",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData { Id = 2, Subject = "Meeting-2",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData { Id = 3, Subject = "Meeting-3",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 appData.Add(new AppointmentData { Id = 4, Subject = "Meeting-4",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0), EndTime = new
 DateTime(2018, 2, 15, 12, 30, 0), IsAllDay = true });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

How to close the editor window manually

You can close the editor window by using `closeEditor` method.

## CSHTML

```

@using Syncfusion.EJ2.Schedule
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings {
 dataSource: [{
 Id: 2,
 Subject: 'Milky Way as Melting pot',
 StartTime: new Date(2022, 2, 5, 20, 0, 0),
 EndTime: new Date(2022, 2, 5, 21, 0, 0)
 }]
 })
 .SelectedDate(new DateTime(2022, 2, 5))
 .Render()
)
</div>
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 if (!args.element.querySelector('#closeEditor')) {
 var btnElement = ej.base.createElement("BUTTON", { id:
'closeEditor' });
 args.element.querySelector('.e-footer-
content').appendChild(btnElement);
 var btnObj = new ej.buttons.Button();
 btnElement.textContent = "Close Editor";
 btnObj.appendTo('#closeEditor');
 btnObj.element.onclick = function () {
 scheduleObj.closeEditor();
 };
 }
 }
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
}

```

```

 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2022,
2, 13, 9, 30, 0), EndTime = new DateTime(2022, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2022, 2, 15, 9, 30, 0), EndTime = new DateTime(2022, 2, 15, 11, 0,
0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

#### How to open the quick info popup manually

You can open the quick info popup in scheduler by using the `openQuickInfoPopup` public method. To open the cell quick info popup, you can pass the cell data as an argument to the method. To open the event quick info popup, you should pass the event data object as an argument to the method.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("Show Cell QuickInfo Popup").Render()
 @Html.EJS().Button("btn2").Content("Show Event QuickInfo
Popup").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings {
 dataSource: [{
 Id: 2,
 Subject: 'Milky Way as Melting pot',
 StartTime: new Date(2022, 2, 5, 20, 0, 0),
 EndTime: new Date(2022, 2, 5, 21, 0, 0)
 }]
 })
 .SelectedDate(new DateTime(2022, 2, 5))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var cellData = {
 Subject: 'Milky Way as Melting pot',
 StartTime: new Date(2022, 2, 5, 20, 0, 0),
 EndTime: new Date(2022, 2, 5, 21, 0, 0)
 }
 }

```

```

 };
 scheduleObj.openQuickInfoPopup(cellData, 'Add');
};
document.getElementById('btn2').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var eventData = {
 Id: 2,
 Subject: 'Milky Way as Melting pot',
 StartTime: new Date(2022, 2, 5, 20, 0, 0),
 EndTime: new Date(2022, 2, 5, 21, 0, 0)
 };
 scheduleObj.openQuickInfoPopup(eventData, 'Save');
};
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2022,
2, 13, 9, 30, 0), EndTime = new DateTime(2022, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2022, 2, 15, 9, 30, 0), EndTime = new DateTime(2022, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### *How to close the quick info popup manually*

You can close the quick info popup in scheduler by using the `closeQuickInfoPopup` public method. The following code example demonstrates the how to close quick info popup manually.



**CSHTML**

```

@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("Close Quick Info Popup").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings {
 dataSource: [{
 Id: 2,
 Subject: 'Milky Way as Melting pot',
 StartTime: new Date(2022, 2, 5, 20, 0, 0),
 EndTime: new Date(2022, 2, 5, 21, 0, 0)
 }]
 })
 .SelectedDate(new DateTime(2022, 2, 5))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
 document.getElementById('schedule').ej2_instances[0];
 scheduleObj.closeQuickInfoPopup();
 };
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2022,
2, 13, 9, 30, 0), EndTime = new DateTime(2022, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData

```

```

 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2022, 2, 15, 9, 30, 0), EndTime = new DateTime(2022, 2, 15, 11, 0,
0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Timezone

The Scheduler makes use of the current system time zone by default. If it needs to follow some other user-specific time zone, then the **Timezone** property needs to be used. Apart from the default action of applying specific timezone to the Scheduler, it is also possible to set different time zone values for each appointments through the properties **StartTimezone** and **EndTimezone** which can be defined as separate fields within the event fields collection.

**Note:** **Timezone** property only applicable for the appointment processing and current time indication.

### Understanding date manipulation in JavaScript

The **new Date()** in JavaScript returns the exact current date object with complete time and timezone information. For example, it may return value such as **Wed Dec 12 2018 05:23:27 GMT+0530 (India Standard Time)** which indicates that the current date is December 12, 2018 and the current time is 5.23 AM on browsers following the IST timezone.

### Scheduler with no timezone

When no specific time zone is set to Scheduler, appointments will be displayed based on the client system's timezone which is the default behavior. Here, the same appointment when viewed from different timezone will have different start and end times.

The following code example displays an appointment from 9.00 AM to 10.00 AM when you open the Scheduler from any of the timezone. This is because, we are providing the start and end time enclosing with **new Date()** which works based on the client browser's timezone.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Scheduler set to specific timezone**

When a time zone is set to Scheduler through **Timezone** property, the appointments will be displayed exactly based on the Scheduler timezone regardless of its client timezone. In the following code example, appointments will be displayed based on Eastern Time (UTC -05:00).

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Timezone("America/New_York")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{

```

```

List<AppointmentData> appData = new List<AppointmentData>();
appData.Add(new AppointmentData
{
 Id = 2,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
});
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

Display events on same time everywhere with no time difference

Setting **Timezone** to UTC for Scheduler will display the appointments on same time as in the database for all the users in different time zone.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Timezone("UTC")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData

```

```
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Set specific timezone for events

It is possible to set different timezone for Scheduler events by setting **StartTimezone** and **EndTimezone** properties within the **EventSettings** option. It allows each appointment to maintain different timezone and displays on Scheduler with appropriate time differences.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 StartTimezone = "Europe/Moscow",
 EndTimezone = "Europe/Moscow"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string StartTimezone { get; set; }
 public string EndTimezone { get; set; }
}
```

Add or remove timezone names to/from the timezone collection

Instead of displaying all the timezone names within the timezone collection (more than 200 are displayed on the editor window timezone fields by default), you can customize the timezone collection at application end as shown in the following example.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 var data = [
 { Value: 'America/New_York', Text: '(UTC-05:00) Eastern Time' },
 { Value: 'UTC', Text: 'UTC' },
 { Value: 'Asia/Kolkata', Text: '(UTC+05:30) India Standard Time' }
];
 var timezoneData = ej.schedule.timezoneData;
 timezoneData.splice.apply(timezoneData, [0,
 timezoneData.length].concat(data));
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0)
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
```

```

public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}

```

## Timezone methods

### offset

This method is used to calculate the difference between passed UTC date and timezone.

| Parameters        | Type   | Description              |
|-------------------|--------|--------------------------|
| ----- ----- ----- |        |                          |
| Date              | Date   | UTC time as date object. |
| Timezone          | String | Timezone.                |

Returns **number**

```

`sh
// Assume your local timezone as IST/UTC+05:30
var timezone = new ej.schedule.Timezone();
var date = new Date(2018,11,5,15,25,11);
var timeZoneOffset = timezone.offset(date,"Europe/Paris");
console.log(timeZoneOffset); //-60
`

```

### convert

This method is used to convert the passed date from one timezone to another timezone.

| Parameters        | Type          | Description                                    |
|-------------------|---------------|------------------------------------------------|
| ----- ----- ----- |               |                                                |
| Date              | Date          | UTC time as date object.                       |
| fromOffset        | number/string | Timezone from which date need to be converted. |
| toOffset          | number/string | Timezone to which date need to be converted.   |

Returns **Date**

```

`sh
// Assume your local timezone as IST/UTC+05:30
var timezone = new ej.schedule.Timezone();
var date = new Date(2018,11,5,15,25,11);
var convertedDate = timezone.convert(date, "Europe/Paris", "Asia/Tokyo");
var convertedDate1 = timezone.convert(date, 60, -360);
console.log(convertedDate); //2018-12-05T17:55:11.000Z
`

```

```
console.log(convertedDate1); //2018-12-05T16:55:11.000Z
```

,

#### *add*

This method is used to add the time difference between passed UTC date and timezone.

| Parameters | Type | Description |
|------------|------|-------------|
|------------|------|-------------|

|       |       |       |
|-------|-------|-------|
| ----- | ----- | ----- |
|-------|-------|-------|

|      |      |                          |
|------|------|--------------------------|
| Date | Date | UTC time as date object. |
|------|------|--------------------------|

|          |        |           |
|----------|--------|-----------|
| Timezone | String | Timezone. |
|----------|--------|-----------|

Returns **Date**

```
`sh
```

```
// Assume your local timezone as IST/UTC+05:30
```

```
var timezone = new ej.schedule.Timezone();
```

```
var date = new Date(2018,11,5,15,25,11);
```

```
var convertedDate = timezone.add(date, "Europe/Paris");
```

```
console.log(convertedDate); //2018-12-05T05:25:11.000Z
```

,

#### *remove*

This method is used to remove the time difference between passed UTC date and timezone.

| Parameters | Type | Description |
|------------|------|-------------|
|------------|------|-------------|

|       |       |       |
|-------|-------|-------|
| ----- | ----- | ----- |
|-------|-------|-------|

|      |      |                     |
|------|------|---------------------|
| Date | Date | UTC as date object. |
|------|------|---------------------|

|          |        |           |
|----------|--------|-----------|
| Timezone | String | Timezone. |
|----------|--------|-----------|

Returns **Date**

```
`sh
```

```
// Assume your local timezone as IST/UTC+05:30
```

```
var timezone = new ej.schedule.Timezone();
```

```
var date = new Date(2018,11,5,15,25,11);
```

```
var convertedDate = timezone.remove(date, "Europe/Paris");
```

```
console.log(convertedDate); //2018-12-05T14:25:11.000Z
```

,

#### *removeLocalOffset*

This method is used to remove the local offset time from the date passed.

| Parameters | Type | Description |
|------------|------|-------------|
|------------|------|-------------|

|       |       |       |
|-------|-------|-------|
| ----- | ----- | ----- |
|-------|-------|-------|



| Date | Date | UTC as date object. |

Returns **Date**

```
`sh
```

```
// Assume your local timezone as IST/UTC+05:30
var timezone = new ej.schedule.Timezone();
var date = new Date(2018,11,5,15,25,11);
var convertedDate = timezone.removeLocalOffset(date);
console.log(convertedDate); //2018-12-05T15:25:11.000Z
`
```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## Views

The Scheduler includes wide variety of view modes with unique configuration options for each view. The available view modes are Day, Week, Work Week, Month, Year, Agenda, Month Agenda, Timeline Day, Timeline Week, Timeline Work Week and Timeline Month, Timeline Year, out of which the **Week** view is set as active.

To navigate between different views and dates, the navigation options are available at the Scheduler header bar. The active view option is usually highlighted by default. The date range of the active view will also be displayed at the left corner of the header bar, clicking on which will open a calendar popup for the ease of desired date selection.

**Note:** By default, Scheduler displays the calendar views such as day, week, work week, month and agenda.

### Setting specific view on scheduler

As the Scheduler displays **Week** view by default, therefore to change the active view, set **currentView** property with the desired view name. The applicable view names that the Scheduler accepts are as follows,

- Day
- Week
- WorkWeek
- Month
- Year
- Agenda
- MonthAgenda
- TimelineDay
- TimelineWeek
- TimelineWorkWeek
- TimelineMonth
- TimelineYear

It is possible to display only the desired views on the Scheduler using the [e-schedule-views](#) property.

In the following example, the Scheduler displays 2 views namely, Week, and TimelineDay.

#### **CSHTML**

```
@using Syncfusion.EJ2.Schedule
@model List<ScheduleView>
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Views(Model)
 .Height("550px")
 .SelectedDate(new DateTime(2022, 2, 15))
 .Render()
)
```

#### **DATA.CS**

```
public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineDay }
 };
 ViewBag.view = viewOption;
 return View();
}
```

To configure Scheduler with different configurations on each view, refer the following code example. Here, the Week view displays the dates in **dd-MM-yyyy** format whereas the Month view hides the weekend days and also displays it in readonly mode.

#### **CSHTML**

```
@using Syncfusion.EJ2.Schedule
@model List<ScheduleView>
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(Model)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

#### **DATA.CS**

```
public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week,
 DateFormat = "dd-MMM-yyyy" },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month,
 ShowWeekend = false, Readonly = true }
 };
 ViewBag.view = viewOption;
 return View();
}
```

```
};
ViewBag.view = viewOption;
return View();
}
```

### View specific configuration

There are scenarios where each view may need to have different configurations. For such cases, you can define the applicable scheduler properties within the `views` Property for each view option as depicted in the following examples. The fields available to be used within each view options are as follows.

| Property                            | Type       | Description                                                                                                                                                                         | Applicable views                                        |
|-------------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------|
| <code>option</code>                 | String     | It accepts the Scheduler view name, based on which we can define its related properties. The view names can be <code>Day</code> , <code>Week</code> and so on.                      | All views.                                              |
| <code>isSelected</code>             | Boolean    | It acts similar to the <code>currentView</code> property and defines the active view of the Scheduler.                                                                              | All views.                                              |
| <code>dateFormat</code>             | Date       | By default, Scheduler follows the date format as per the default culture assigned to it. When it is defined under specific view, only those assigned views follow this date format. | All views.                                              |
| <code>readonly</code>               | Boolean    | When set to <code>true</code> , prevents the CRUD actions on the respective view under where it is defined.                                                                         | All views.                                              |
| <code>resourceHeaderTemplate</code> | String     | The template option which is used to customize the resource header cells on the Scheduler. It gets applied only on the views, wherever it is defined.                               | All views.                                              |
| <code>dateHeaderTemplate</code>     | String     | The template option which is used to customize the date header cells and is applied only on the views, wherever it is defined.                                                      | All views.                                              |
| <code>eventTemplate</code>          | String     | The template option to customize the events background. It will get applied to the events of the view to which it is currently being defined.                                       | All views.                                              |
| <code>showWeekend</code>            | Boolean    | When set to <code>false</code> , it hides the weekend days of a week from the views on which it is defined.                                                                         | All views.                                              |
| <code>group</code>                  | GroupModel | Allows to set different resource grouping options on all available Scheduler view modes.                                                                                            | All views.                                              |
| <code>cellTemplate</code>           | String     | The template option to customize the work cells of the Scheduler and is applied only on the views, on which it is defined.                                                          | Applicable on all views except Agenda view.             |
| <code>workDays</code>               | Number[]   | It is used to set the working days on the Scheduler views.                                                                                                                          | Applicable on all views except Agenda view.             |
| <code>displayName</code>            | String     | When a particular view is customized to display with different intervals, this property allows the user to set different display name for each of the views.                        | Applicable on all views except Agenda and Month Agenda. |
| <code>interval</code>               | Number     | It allows to customize the default Scheduler views with different set of days, weeks, work weeks or months on the applicable view type.                                             | Applicable on all views except Agenda and Month Agenda. |

| **startHour** | String | It is used to specify the start hour, from which the Scheduler should be displayed. It accepts the time string in a short skeleton format and also, hides the time beyond the specified start time. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. |

| **endHour** | String | It is used to specify the end hour, at which the Scheduler ends. It accepts the time string in a short skeleton format. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| **timeScale** | [TimeScaleModel](#) | Allows to set different timescale configuration on each applicable view modes. | Applicable on Day, Week, Work Week, Timeline Day, Timeline Week, and Timeline Work Week views. |

| **showWeekNumber** | Boolean | When set to **true**, shows the week number on the respective weeks. | Applicable on Day, Week, Work Week, and Month views. |

| **allowVirtualScrolling** | Boolean | It is used to enable or disable the virtual scrolling functionality. | Applicable on Agenda and Timeline views. |

| **headerRows** | [HeaderRowsModel](#) | Allows defining the custom header rows on timeline views of the Scheduler to display the year, month, week, date and hour label as an individual row. | Applicable only on all timeline views. |

#### Day view

Usually a day view displays a single day with all its related appointments. It is possible to customize the day view to display more number of days by extending the **e-schedule-views** property with **interval** option. You can also define any of the above defined properties within the **e-schedule-views** object definition as depicted in the following code example.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).DisplayName("3 Days").Interval(3).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 13))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
```

```

 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 3, 10, 10, 0, 0), EndTime = new DateTime(2023, 3, 10, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 3, 11, 9, 30, 0), EndTime = new DateTime(2023, 3, 11, 12, 0, 0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

**Note:** All the above defined properties can be accessed within Day view except `allowVirtualScrolling` and `headerRows`.

#### Week view

The Week view displays a count of 7 days (from Sunday to Saturday) with all its related appointments. The first day of the week can be changed using the `firstDayOfWeek` which accepts the integer (Sunday=0, Monday=1, Tuesday=2 and so on) value. You can navigate to a particular date in day view from the week view by clicking on the appropriate dates on the date header bar.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Week).DisplayName("2 Weeks").Interval(2).IsSelected(true).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource = ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 1, 19, 10, 0, 0), EndTime = new DateTime(2023, 1, 19, 12, 0, 0) });
 appData.Add(new AppointmentData

```

```

 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 1, 21,
9, 30, 0) , EndTime = new DateTime(2023, 1, 21, 12, 0, 0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

**Note:** All the above defined properties in the table can be accessed within Week and Work week views except `allowVirtualScrolling` and `headerRows`.

#### Work Week view

The Work week view displays only the working days of a week (count of 5 days) and its associated appointments. It is possible to customize the working days on the work week view by using the `workDays` property which accepts an array of integer values (such as Sunday=0, Monday=1, Tuesday=2 and so on). By default, it displays from Monday to Friday (5 days). You can also navigate to a particular date in the day view from the work week view by clicking on the appropriate dates in the date header bar.

The following code example depicts how to change the working days only on the **Work Week** view of the Scheduler.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.WorkWeek).WorkDays(ViewBag.workDays).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 14))
 .Render()
)

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workDays = new int[] { 1, 3, 5 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData

```

```

 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 1, 17, 10, 0, 0) , EndTime = new DateTime(2023, 1, 17, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 1, 19, 9, 30, 0) , EndTime = new DateTime(2023, 1, 19, 12, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** The Week, Work week and Day views can display the all-day row appointments in a separate all-day row with an expand/collapse option to view it.

#### Month view

A Month view displays the entire days of a particular month and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the month cells.

By default, when you try to create an appointment through Month view, it is considered as created for an entire day. You can explicitly change this behavior by unchecking the **All-day** option from editor window, so that it defaults to the start time duration as 9.00 AM and end time as 9.30 AM.

You can also have the **+ more** text indicator on each day cell of a Month view, clicking on which will allow you to view the hidden appointments of a day.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.ReadOnly(true).Option(View.Month).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
}

```

```

 appData.Add(new AppointmentData
 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 2, 13, 10, 0, 0) , EndTime = new DateTime(2023, 2, 13, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 2, 17, 9, 30, 0) , EndTime = new DateTime(2023, 2, 17, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Seminar", StartTime = new DateTime(2023, 2, 22, 14, 30, 0) , EndTime = new DateTime(2023, 2, 22, 18, 0, 0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

### Year view

A Year view displays all the days of a particular year with months and all its related appointments. You can navigate to a particular date in the day view by clicking on the appropriate date text on the year cells.

Year view is available in both the **Horizontal** and **Vertical** orientations. You can manage the orientation of year view through **views** property.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.ReadOnly(true).Option(View.Year).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData

```



```

 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 3, 4, 0, 0, 0) , EndTime = new DateTime(2023, 3, 5, 0, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 5, 1, 9, 30, 0) , EndTime = new DateTime(2023, 5, 1, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Seminar", StartTime = new DateTime(2023, 1, 2, 9, 30, 0) , EndTime = new DateTime(2023, 1, 2, 12, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** The year view also has module support. In that, you can get all the months of a particular year in a calendar view format. In that calendar view, appointment contained dates are highlighted with dots placed under the individual date. When you click on the date, the event popup will be displayed and the events will be listed.

#### Agenda view

The Agenda view lists out the appointments in a grid-like view for the next 7 days by default from the current date. The count of the days can be changed using the API `agendaDaysCount`. It allows virtual scrolling of dates by enabling the `allowVirtualScrolling` property. Also, you can enable or disable the display of days on Scheduler that has no appointments by setting true or false to the `hideEmptyAgendaDays` property.

The following code example depicts how to customize the display of events within Agenda view alone.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view =>
 {
 view.Option(View.Agenda)
 .EventTemplate("<div class='template-wrap'><div class='subject'>${Subject}</div></div>")
 .AllowVirtualScrolling(false)
 .Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource = ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

#### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 2, 13, 10, 0, 0), EndTime = new DateTime(2023, 2, 13, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 12, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** Schedule Height is mandatory to set in pixels for Agenda view alone.

#### *Month Agenda view*

A Month-Agenda view shows a month calendar, where clicking on a particular day will display the appointments present on that date below the calendar. The day with appointments are differentiated with a circular dot below the date of the calendar.

The following code example shows how to hide the weekend days on **MonthAgenda** view as well as the working days list is modified on Month Agenda view alone.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {

view.Option(View.MonthAgenda).WorkDays(ViewBag.workDays).ShowWeekend(false).
Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()

```

```

{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workDays = new int[] { 0, 3, 6 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 2, 12, 10, 0, 0), EndTime = new DateTime(2023, 2, 12, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 2, 12, 9, 30, 0), EndTime = new DateTime(2023, 2, 12, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Seminar", StartTime = new DateTime(2023, 2, 22, 14, 30, 0), EndTime = new DateTime(2023, 2, 22, 18, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### *Timeline views – Day, Week, Work Week*

Similar to the day view, timeline day view shows a single day with all its appointments where the time slots are displayed horizontally. By default, the cell height adjusts as per the height set to Scheduler.

When the number of appointments exceeds the visible area of the cells, the **+ more** text indicator will be displayed at the bottom to denote the presence of few more appointments in that time range.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {

view.Option(View.TimelineDay).StartHour("08:00").EndHour("20:00").Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
}

```

```

 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 13, 0, 0), EndTime = new DateTime(2023, 2, 15, 14, 0,
0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

Similar to the Week view, the timeline week view shows 7 days with its associated appointments with the time slots displayed horizontally.

### C#HTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.TimelineWeek).TimeScale(ts =>
ts.Enable(false)).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

 public ActionResult Index()
 {
 ViewBag.datasource = GetScheduleData();
 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 9, 30, 0), EndTime = new DateTime(2023, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData

```

```

 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

The following code example depicts how to display the timeline work week view on Scheduler,

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.TimelineWorkWeek).WorkDays(ViewBag.workDays).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workDays = new int[] { 1, 3, 5 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 12, 9, 30, 0), EndTime = new DateTime(2023, 2, 12, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
}

```

```

 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** Clicking on the dates in the date header bar of Timeline day, Timeline week and Timeline work week will allow you to navigate to the Agenda view.

#### *Timeline Month view*

A Timeline Month view displays the current month days along with its appointments.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.TimelineMonth).ShowWeekend(false).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workDays = new int[] { 1, 2, 3 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 2, 10, 10, 0, 0), EndTime = new DateTime(2023, 2, 10, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 2, 10, 9, 30, 0), EndTime = new DateTime(2023, 2, 10, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Seminar", StartTime = new DateTime(2023, 2, 22, 14, 30, 0), EndTime = new DateTime(2023, 2, 22, 18, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** Clicking on the dates in the date header bar of Timeline month will allow you to navigate to the Timeline day view.

#### Timeline Year view

In Timeline Year view, each row depicts a single resource. Whereas in the vertical view, each resource is grouped parallelly as columns. Here, the resource grouping follows the tree-view like hierarchical grouping structure and can contain any level of child resources.

To make use of the timeline Year view on Scheduler, import and inject `TimelineYear` module from the `ej2-schedule` package.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.TimelineYear).ShowWeekend(false).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workDays = new int[] { 1, 2, 3 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Meeting", StartTime = new DateTime(2023, 3, 4, 0, 0, 0), EndTime = new DateTime(2023, 3, 5, 0, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Conference", StartTime = new DateTime(2023, 5, 1, 9, 30, 0), EndTime = new DateTime(2023, 5, 1, 12, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Seminar", StartTime = new DateTime(2023, 1, 2, 9, 30, 0), EndTime = new DateTime(2023, 1, 2, 12, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Resource grouping

The following code example depicts how to group the multiple resources on Timeline Year view and its relevant events are displayed accordingly under those resources.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

res.AllowMultiple(false).DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").ColorField("RoomColor").Add();

res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.TimelineYear).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasource))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 // datasource for room resources
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" });
 rooms.Add(new RoomResource { RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" });
 ViewBag.Rooms = rooms;
 // datasource for owner resources
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor = "#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor = "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3, OwnerGroupId = 1, OwnerColor = "#7499e1" });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Rooms", "Owners" };
 return View();
}

public List<ResourceData> GetResourceData()
{

```



```

List<ResourceData> resourceData = new List<ResourceData>();
resourceData.Add(new ResourceData
{
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 1,
 RoomId = 1
});
resourceData.Add(new ResourceData
{
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 3,
 RoomId = 1
});
resourceData.Add(new ResourceData
{
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 2,
 RoomId = 2
});
return resourceData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
 public int RoomId { get; set; }
}
public class RoomResource
{
 public string RoomText { set; get; }
 public int Id { set; get; }
 public string RoomColor { set; get; }
}
public class OwnerResource
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
 public int OwnerGroupId { set; get; }
}

```

### Auto row height

Timeline Year view supports Auto row height. When the feature `rowAutoHeight` is enabled, the row height gets auto-adjusted based on the number of overlapping events occupied in the same time range. If you disable the Auto row height, you have the `+ more` text indicator on each day cell of a Timeline Year view, clicking on which will allow you to view the hidden appointments of a day.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .RowAutoHeight(true)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Seminar",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false
 });
 appData.Add(new AppointmentData
 {
 Id = 3,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false
 });
 appData.Add(new AppointmentData
 {
```

```

 Id = 4,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false
 });
 return appData;
}
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

### Extending view intervals

It is possible to customize the display of default number of days on different Scheduler view modes. For example, a day view can be extended to display 3 days by setting the `interval` option as 3 for the `Day` option within the `e-schedule-views` property as depicted in the following code example. In the same way, you can also display 2 weeks by setting interval 2 for the `Week` option.

You can provide the alternative display name for such customized views on the Scheduler header bar, by setting the appropriate `displayName` property.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).DisplayName("3 Days").Interval(3).Add();
 view.Option(View.Week).DisplayName("2
Weeks").Interval(2).IsSelected(true).Add();
 view.Option(View.Month).DisplayName("4 Months").Interval(4).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
}

```

```

 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 9, 30, 0), EndTime = new DateTime(2023, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

**Note:** The view intervals can be extended on all the Scheduler view modes except Agenda and Month-Agenda views.

See Also

- [How to restrict view navigation while clicking on dates](#)

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## Calendar mode

The Scheduler supports the following two types of calendar mode.

- Gregorian Calendar
- Islamic Calendar

### Gregorian Calendar

The Scheduler by default displays the gregorian calendar dates which is the most widely used calendar in the world. The Gregorian calendar is a solar calendar which has 12 months with 28 to 31 days each. A year which is divisible by four is said to be a leap year in this calendar mode. A leap year usually has 366 days whereas the regular year has 365 days.

### Islamic Calendar

The Islamic Calendar, also known as the Hijri or Muslim calendar, is a lunar calendar which has 12 months in a year with 354 or 355 days. Each month of this calendar denotes the birth of the new lunar cycle and therefore, each month can have 29 or 30 days depending on the visibility of the moon. Here, the odd-numbered months have 30 days and the even months have 29 days.

**Note:** The current Islamic year is 1440 AH. Usually the Gregorian calendar runs from approximately 11 September 2018 to 30 August 2019 for this 1440 AH year.

The Scheduler has a property `CalendarMode` which is used to switch between the gregorian and islamic calendar modes. By default, it is set to `Gregorian` and to use it with Islamic calendar dates,

define the **CalendarMode** of Scheduler to **Islamic**. The following example depicts, how to display the Islamic calendar dates on Scheduler.

It requires the following CLDR data to be loaded using loadCldr function.

- numberingSystems.json
- ca-gregorian.json
- numbers.json
- timeZoneNames.json
- ca-islamic.json

**Note:** To know more information on, how to install the CLDR data, refer the [Internationalization](#) topic.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .CalendarMode(CalendarType.Islamic)
 .EnableRtl(true)
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script>
 document.addEventListener('DOMContentLoaded', function () {
 var scheduleObject =
 document.getElementById('schedule').ej2_instances[0];
 loadCultureFiles('de');
 scheduleObject.locale = 'de';
 });
 function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numberingSystems.json',
 'numbers.json', 'timeZoneNames.json', 'ca-islamic.json'];
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;
 if (files[prop] === 'numberingSystems.json') {
 ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-
 data/supplemental/' + files[prop], 'GET', false);
 } else {
 ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-
 data/main/' + name + '/' + files[prop], 'GET', false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
 };
 for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
 }
 }
}
```

```
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2018, 2, 14, 9, 30, 0), EndTime = new DateTime(2018, 2, 14, 11, 0,
 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", StartTime = new
 DateTime(2018, 2, 15, 12, 0, 0), EndTime = new DateTime(2018, 2, 15, 14, 0,
 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
 2, 16, 9, 30, 0), EndTime = new DateTime(2018, 2, 16, 11, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

**Note:** However, this feature does not yet support recurrence options, which we are planning to add them in the next release.

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Multiple resources and grouping

Resources and grouping support allows the Scheduler to be shared by multiple resources. Also, the appointments of each resource are displayed under relevant resources. Each resource in the Scheduler is arranged in a column/row wise order, with individual spacing to display all its respective appointments on a single page. It also supports the multiple levels of grouping of resources, thus enabling the categorization of resources in a hierarchical structure and shows it either in expandable groups (Timeline views) or else vertical hierarchy one after the other (Calendar views).

It is also possible to assign one or more resources to the same appointment, by allowing multiple selection of resource options available in the event editor window.

The Scheduler groups the resources based on different criteria. It includes grouping appointments based on resources, grouping resources based on dates, and timeline scheduling. Also, the data for resources bind with Scheduler either as a local JSON collection or URL, retrieving data from remote data services.

### Resource fields

The default options available within the **Resources** collection are as follows,

| Field name     | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field          | String  | A value that binds to the resource field of event object.                                                                                                                                                                                                                                                                                                                                              |
| Title          | String  | It holds the title of the resource field to be displayed on the event editor window.                                                                                                                                                                                                                                                                                                                   |
| Name           | String  | A unique resource name used for differentiating various resource objects while grouping.                                                                                                                                                                                                                                                                                                               |
| AllowMultiple  | Boolean | When set to <b>true</b> , allows multiple selection of resource names, thus creating multiple instances of same appointment for the selected resources.                                                                                                                                                                                                                                                |
| DataSource     | Object  | Assigns the resource <b>DataSource</b> , where data can be passed either as an array of JavaScript objects, or else can create an instance of <a href="#">DataManager</a> in case of processing remote data and can be assigned to the <b>DataSource</b> property. With the remote data assigned to <b>DataSource</b> , check the available <a href="#">adaptors</a> to customize the data processing. |
| Query          | Query   | Defines the external <a href="#">Query</a> that will be executed along with the data processing.                                                                                                                                                                                                                                                                                                       |
| IdField        | String  | Binds the resource ID field name from the resources <b>DataSource</b> .                                                                                                                                                                                                                                                                                                                                |
| ExpandedField  | String  | Binds the <b>ExpandedField</b> name from the resources <b>DataSource</b> . It usually holds boolean value which decide whether the resource of timeline views is in collapse or expand state on initial load.                                                                                                                                                                                          |
| TextField      | String  | Binds the text field name from the resources <b>DataSource</b> . It usually holds the resource names.                                                                                                                                                                                                                                                                                                  |
| GroupIDField   | String  | Binds the group ID field name from the resource <b>DataSource</b> . It usually holds the value of resource IDs of parent level resources.                                                                                                                                                                                                                                                              |
| ColorField     | String  | Binds the color field name from the resource <b>DataSource</b> . The color value mapped in this field will be applied to the events of resources.                                                                                                                                                                                                                                                      |
| StartHourField | String  | Binds the start hour field name from the resource <b>DataSource</b> . It allows to provide different work start hour for the resources.                                                                                                                                                                                                                                                                |
| EndHourField   | String  | Binds the end hour field name from the resource <b>DataSource</b> . It allows to provide different work end hour for the resources.                                                                                                                                                                                                                                                                    |
| WorkDaysField  | String  | Binds the work days field name from the resources <b>DataSource</b> . It allows to provide different working days collection for the resources.                                                                                                                                                                                                                                                        |
| CssClassField  | String  | Binds the custom CSS class field name from the resources <b>DataSource</b> . It maps the CSS class written for the specific resources and applies it to the events of those resources.                                                                                                                                                                                                                 |

### Resource data binding

The data for resources can bind with Scheduler either as a local JSON collection or a service URL, retrieving resource data from remote data services.

#### Using local JSON data

The following code example depicts how to bind the local JSON data to the **DataSource** of **Resources** collection.

- Give the resource datasource in Index method
- Add the Scheduler code in View page

### CSHTML

```
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => { res.DataSource(ViewBag.Owners)
 .Field("OwnerId")
 .Title("Owners")
 .Name("Owners")
 .TextField("text")
 .IdField("id")
 .ColorField("color")
 .Add();
 })
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 string[] resources = new string[] { "Owners" };
 ViewBag.Resources = resources;
 List<OwnerRes> owners = new List<OwnerRes>();
 owners.Add(new OwnerRes { text = "Nancy", id = 1, color = "#ffaa00" });
 owners.Add(new OwnerRes { text = "Steven", id = 2, color = "#f8a398" });
 owners.Add(new OwnerRes { text = "Michael", id = 3, color = "#7499e1" });
});
ViewBag.Owners = owners;
return View();
}

public class OwnerRes
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}
```

#### Using remote service URL

The following code example depicts how to bind the remote data for resources **dataSource**.



- Give the resource datasource in Index method
- Add the Scheduler code in View page

**CSHTML**

```
@(Html.EJS().Schedule("schedule")
.Width("100%")
.Height("550px")
.Group(group => group.Resources(ViewBag.Resources))
.Resources(res => {
 res.AllowMultiple(true)
 .DataSource(d =>
 d.Url("Home/GetResourceData")
 .Adaptor("UrlAdaptor")
 .CrossDomain(true)
)
 .Field("OwnerId")
 .Title("Owner")
 .Name("Owners")
 .TextField("text")
 .IdField("id")
 .ColorField("color")
 .Add();
})
.SelectedDate(new DateTime(2018, 4, 1))
.Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 string[] resources = new string[] { "Owners" };
 ViewBag.Resources = resources;
 return View();
}

public JsonResult GetResourceData()
{
 List<OwnerRes> owners = new List<OwnerRes>();
 owners.Add(new OwnerRes { text = "Nancy", id = 1, color = "#ffaa00" });
 owners.Add(new OwnerRes { text = "Steven", id = 2, color = "#f8a398" });
 owners.Add(new OwnerRes { text = "Michael", id = 3, color = "#7499e1" });
 return Json(owners);
}

public class OwnerRes
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}
```

### Scheduler with multiple resources

It is possible to display the Scheduler in default mode without visually showcasing all the resources in it, but allowing to assign the required resources to the appointments through the event editor resource options.

The appointments belonging to the different resources will be displayed altogether on the default Scheduler, which will be differentiated based on the resource color assigned in the **Resources** (depicting to which resource that particular appointment belongs) collection.

**Example:** To display default Scheduler with multiple resource options in the event editor, ignore the group option and simply define the **Resources** property with all its internal options.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS() .Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Resources(res => {
 res.AllowMultiple(true)
 .DataSource(ViewBag.Owners)
 .Field("OwnerId")
 .Title("Owner")
 .Name("Owners")
 .TextField("OwnerText")
 .IdField("Id")
 .ColorField("OwnerColor")
 .Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 List<OwnerResources> owners = new List<OwnerResources>();
 owners.Add(new OwnerResources { OwnerText = "Nancy", Id = 1, OwnerColor
= "#ffaa00" });
 owners.Add(new OwnerResources { OwnerText = "Steven", Id = 2, OwnerColor
= "#f8a398" });
 owners.Add(new OwnerResources { OwnerText = "Michael", Id = 3,
OwnerColor = "#7499e1" });
 ViewBag.Owners = owners;
 return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
```

```

 Subject = "Workflow Analysis",
 StartTime = new DateTime(2023, 4, 3, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 3, 13, 0, 0),
 IsAllDay = false,
 OwnerId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 4, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 4, 13, 0, 0),
 IsAllDay = false,
 OwnerId = 2
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 5, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 5, 13, 0, 0),
 IsAllDay = false,
 OwnerId = 3
 });
 return resourceData;
}
public class OwnerResources
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
}

```

**Note:** Setting `AllowMultiple` to `true` in the above code example allows you to select multiple resources from the event editor and also creates multiple copies of the same appointment in the Scheduler for each resources while rendering.

### Resource grouping

Resource grouping support allows the Scheduler to group the resources in a hierarchical structure both as an expandable groups (Timeline views) and as vertical hierarchy displaying resources one after the other (Resources view).

Scheduler supports both single and multiple levels of resource grouping that can be customized both in timeline and vertical Scheduler views.

*Vertical resource view*

The following code example displays how the multiple resources are grouped and its events are portrayed in the default calendar views.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.ByGroupID(false).Resources(ViewBag.Resources))
 .Resources(res => {
 res.DataSource(ViewBag.Projects).Field("ProjectId").Title("Choose
Project").Name("Projects").TextField("text").IdField("id").ColorField("color")
 }.Add());

res.AllowMultiple(true).DataSource(ViewBag.Categories).Field("CategoryId").Title("Category").Name("Categories").TextField("text").IdField("id").ColorField("color").Add();
 })
 .Views(view => {
 view.Option(View.Week).Add();
 view.Option(View.Month).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetTimelineResourceData();
 // datasource for project resources
 List<ResourceFields> projects = new List<ResourceFields>();
 projects.Add(new ResourceFields { text = "PROJECT 1", id = 1, color = "#cb6bb2" });
 projects.Add(new ResourceFields { text = "PROJECT 2", id = 2, color = "#56ca85" });
 projects.Add(new ResourceFields { text = "PROJECT 3", id = 3, color = "#df5286" });
 ViewBag.Projects = projects;
 // datasource for category resources
 List<ResourceFields> categories = new List<ResourceFields>();
 categories.Add(new ResourceFields { text = "Development", id = 1, color = "#df5286" });
 categories.Add(new ResourceFields { text = "Testing", id = 2, color = "#7fa900" });
 ViewBag.Categories = categories;
 ViewBag.Resources = new string[] { "Projects", "Categories" };
 return View();
}

public List<ResourceData> GetTimelineResourceData()
```

```
{
 List<ResourceData> timelineResourceData = new List<ResourceData>();
 timelineResourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Decoding",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 1,
 CategoryId = 1
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Bug Automation",
 StartTime = new DateTime(2023, 4, 4, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 4, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 2,
 CategoryId = 1
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Functionality testing",
 StartTime = new DateTime(2023, 4, 5, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 5, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 3,
 CategoryId = 1
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 4,
 Subject = "Resolution-based testing",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 1,
 CategoryId = 2
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 5,
 Subject = "Test report Validation",
 StartTime = new DateTime(2023, 4, 4, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 4, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 2,
 CategoryId = 2
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 6,
 Subject = "Test case correction",
 StartTime = new DateTime(2023, 4, 5, 9, 30, 0),
```

```

 EndTime = new DateTime(2023, 4, 5, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 3,
 CategoryId = 2
 });
 return timelineResourceData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int ProjectId { get; set; }
 public int CategoryId { get; set; }
}
public class ResourceFields
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}

```

#### *Timeline resource view*

The following code example depicts how to group the multiple resources on Timeline Scheduler views and its relevant events are displayed accordingly under those resources.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

 res.AllowMultiple(false).DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").ColorField("RoomColor").Add();

 res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)

```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 // datasource for room resources
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { RoomText = "ROOM 1", Id = 1, RoomColor =
"#cb6bb2" });
 rooms.Add(new RoomResource { RoomText = "ROOM 2", Id = 2, RoomColor =
"#56ca85" });
 ViewBag.Rooms = rooms;
 // datasource for owner resources
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerGroupId
= 1, OwnerColor = "#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2,
OwnerGroupId = 2, OwnerColor = "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3,
OwnerGroupId = 1, OwnerColor = "#7499e1" });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Rooms", "Owners" };
 return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 1,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 3,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 2,
 RoomId = 2
 });
}
```

```

 return resourceData;
 }
 public class ResourceData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
 public int RoomId { get; set; }
 }
 public class RoomResource
 {
 public string RoomText { set; get; }
 public int Id { set; get; }
 public string RoomColor { set; get; }
 }
 public class OwnerResource
 {
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
 public int OwnerGroupId { set; get; }
 }
}

```

#### Grouping single-level resources

This kind of grouping allows the Scheduler to display all the resources at a single level simultaneously. The appointments mapped under resources will be displayed with the colors as per the **ColorField** defined on the resources collection.

**Example:** To display the Scheduler with single level resource grouping,

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

 res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner")
 .Name("Owners").TextField("OwnerText").IdField("Id").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.Week).Add();
 view.Option(View.Month).Add();
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)

```



```
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerColor =
"#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2, OwnerColor
= "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3, OwnerColor
= "#7499e1" });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Owners" };
 return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 3, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 3, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 4, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 4, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 2
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2023, 4, 5, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 5, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 3
 });
 return resourceData;
}

public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
```

```

 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
}
public class OwnerResource
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
}

```

**Note:** The **Name** field defined in the **resources** collection namely **Owners** will be mapped within the **Group** property, in order to enable the grouping option with those resource levels on the Scheduler.

#### *Grouping multi-level resources*

It is possible to group the resources of Scheduler in multiple levels, by mapping the child resources to each parent resource. In the following example, there are 2 levels of resources, on which the second level resources are defined with **GroupIDField** mapping to the first level resource's ID so as to establish the parent-child relationship between them.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

res.AllowMultiple(false).DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").ColorField("RoomColor").Add();

res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasource))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)

```

#### **DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 // datasource for room resources
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" });
}

```

```

rooms.Add(new RoomResource { RoomText = "ROOM 2", Id = 2, RoomColor =
"#56ca85" });
ViewBag.Rooms = rooms;
// datasource for owner resources
List<OwnerResource> owners = new List<OwnerResource>();
owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerGroupId
= 1, OwnerColor = "#ffaa00" });
owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2,
OwnerGroupId = 2, OwnerColor = "#f8a398" });
owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3,
OwnerGroupId = 1, OwnerColor = "#7499e1" });
ViewBag.Owners = owners;
ViewBag.Resources = new string[] { "Rooms", "Owners" };
return View();
}
public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 1,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 3,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 2,
 RoomId = 2
 });
 return resourceData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

```

 public int OwnerId { get; set; }
 public int RoomId { get; set; }
}
public class RoomResource
{
 public string RoomText { set; get; }
 public int Id { set; get; }
 public string RoomColor { set; get; }
}
public class OwnerResource
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
 public int OwnerGroupId { set; get; }
}

```

### One-to-One grouping

In multi-level grouping, Scheduler usually groups the resources on the child level based on the **GroupIdField** that maps with the **IdField** field of parent level resources (as **ByGroupId** set to true by default). There are also option which allows you to group all the child resource(s) against each of its parent resource(s). To enable this kind of grouping, set **false** to the **ByGroupId** option within the **Group** property. In the following code example, there are two levels of resources, on which all the 3 resources at the child level is mapped one to one with each resource on the first level.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .CurrentView(View.WorkWeek)
 .Group(group => group.ByGroupId(false).Resources(ViewBag.Resources))
 .Resources(res =>
 {
 res.DataSource(ViewBag.Projects).Field("ProjectId").Title("Choose Project").Name("Projects").TextField("text").IdField("id").ColorField("color").Add();

 res.DataSource(ViewBag.Categories).Field("CategoryId").Title("Category").Name("Categories").TextField("text").IdField("id").GroupIdField("groupId").ColorField("color").AllowMultiple(true).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetResourceTeamData();
 // datasource for project resources
 List<ProjectResource> projects = new List<ProjectResource>();
}

```

```

 projects.Add(new ProjectResource { text = "PROJECT 1", id = 1, color =
"#cb6bb2" });
 projects.Add(new ProjectResource { text = "PROJECT 2", id = 2, color =
"#56ca85" });
 ViewBag.Projects = projects;
 // datasource for category resources
 List<CategoryResource> categories = new List<CategoryResource>();
 categories.Add(new CategoryResource { text = "Development", id = 1,
groupId = 1, color = "#1aaa55" });
 categories.Add(new CategoryResource { text = "Testing", id = 2, groupId
= 2, color = "#7fa900" });
 categories.Add(new CategoryResource { text = "Documentation", id = 3,
groupId = 2, color = "#7499e1" });
 ViewBag.Categories = categories;
 ViewBag.Resources = new string[] { "Projects", "Categories" };
 return View();
}
public List<ResourceTeamData> GetResourceTeamData()
{
 List<ResourceTeamData> resourceTeamData = new List<ResourceTeamData>();
 resourceTeamData.Add(new ResourceTeamData
 {
 Id = 1,
 Subject = "Developers Meeting",
 StartTime = new DateTime(2023, 4, 3, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 3, 12, 0, 0),
 RecurrenceRule = "FREQ=WEEKLY; INTERVAL=1; BYDAY=MO, TU, WE, TH, FR",
 ProjectId = 1,
 CategoryId = 1
 });
 resourceTeamData.Add(new ResourceTeamData
 {
 Id = 2,
 Subject = "Test report Validation",
 StartTime = new DateTime(2023, 4, 4, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 4, 12, 0, 0),
 RecurrenceRule = "FREQ=WEEKLY; INTERVAL=1; BYDAY=MO, WE, FR",
 ProjectId = 1,
 CategoryId = 3
 });
 resourceTeamData.Add(new ResourceTeamData
 {
 Id = 3,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 5, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 5, 12, 0, 0),
 RecurrenceRule = "FREQ=WEEKLY; INTERVAL=1; BYDAY=MO, TU, WE, TH, FR",
 ProjectId = 2,
 CategoryId = 2
 });
 return resourceTeamData;
}
public class ProjectResource
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}

```

```

}
public class CategoryResource
{
 public string text { set; get; }
 public int id { set; get; }
 public int groupId { set; get; }
 public string color { set; get; }
}
public class ResourceTeamData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public int ProjectId { get; set; }
 public int CategoryId { get; set; }
 public string RecurrenceRule { get; set; }
}

```

### Grouping resources by date

It groups the number of resources under each date and is applicable only on the calendar views such as Day, Week, Work Week, Month, Agenda and Month-Agenda. To enable such grouping, set **ByDate** option to **true** within the **Group** property.

**Example:** To display the Scheduler with resources grouped by date,

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.ByDate(true).Resources(ViewBag.Resources))
 .Resources(res => {

res.DataSource(ViewBag.Owners).Field("OwnerId").Title("Assignee").Name("Owners")
.TextField("text").IdField("id").ColorField("color").AllowMultiple(true)
.Add();
 })
 .Views(view => {
 view.Option(View.Week).Add();
 view.Option(View.Month).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasources = GetResourceData();
 List<OwnerResource> owners = new List<OwnerResource>();
}

```

```

 owners.Add(new OwnerResource { text = "Alice", id = 1, color = "#ffaa00"
 });
 owners.Add(new OwnerResource { text = "Smith", id = 2, color = "#f8a398"
 });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Owners" };
 return View();
}
public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 3, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 3, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 4, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 4, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 2
 });
 return resourceData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
}
public class OwnerResource
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}

```

**Note:** This kind of grouping by date is not applicable on any of the **timeline views**.

#### Customizing parent resource cells

In timeline view work cells of parent resource can be customized by checking the `elementType` as `resourceGroupCells` in the event `RenderCell`. In the following code example, background color of the work hours has been changed.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .RenderCell("onRender")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res =>
 {

res.AllowMultiple(false).DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").ColorField("RoomColor").Add();

res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").Add();
 })
 .Views(view =>
 {
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasource))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)
<script type="text/javascript">
 function onRender(args) {
 if (args.elementType == 'resourceGroupCells' &&
args.element.className.indexOf('e-work-hours') > 0) {
 args.element.style.background = "#FAFAE3";
 }
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 // datasource for room resources
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2" });
 rooms.Add(new RoomResource { RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85" });
 ViewBag.Rooms = rooms;
 // datasource for owner resources
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor = "#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor = "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3, OwnerGroupId = 1, OwnerColor = "#7499e1" });
}

```



```

ViewBag.Owners = owners;
ViewBag.Resources = new string[] { "Rooms", "Owners" };
return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 1, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 1,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 4, 1, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 3,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2023, 4, 1, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 2,
 RoomId = 2
 });
 return resourceData;
}

public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
 public int RoomId { get; set; }
}

public class RoomResource
{
 public string RoomText { set; get; }
 public int Id { set; get; }
 public string RoomColor { set; get; }
}

public class OwnerResource
{

```

```

public string OwnerText { set; get; }
public int Id { set; get; }
public string OwnerColor { set; get; }
public int OwnerGroupId { set; get; }
}

```

### Working with shared events

Multiple resources can share the same events, thus allowing the CRUD action made on it to reflect on all other shared instances simultaneously. To enable such option, set `AllowGroupEdit` option to `true` within the `Group` property. With this property enabled, a single appointment object will be maintained within the appointment collection, even if it is shared by more than one resource – whereas the resource fields of such appointment object will be in array which hold the IDs of the multiple resources.

**Note:** Any actions such as create, edit or delete held on any one of the shared event instances, will be reflected on all other related instances visible on the UI.

**Example:** To edit all the resource events simultaneously,

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.AllowGroupEdit(true).Resources(ViewBag.Resources))
 .Resources(res => {

res.DataSource(ViewBag.Conferences).Field("ConferenceId").Title("Attendees")
.Name("Conferences").TextField("text").IdField("id").ColorField("color").All
owMultiple(true).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasource))
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 14))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineWeek },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineMonth },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Agenda }
 };
 ViewBag.view = viewOption;
 List<ConferenceRes> conferences = new List<ConferenceRes>();
}

```

```

 conferences.Add(new ConferenceRes { text = "Margaret", id = 1, color =
"#1aaa55" });
 conferences.Add(new ConferenceRes { text = "Robert", id = 2, color =
"#357cd2" });
 conferences.Add(new ConferenceRes { text = "Laura", id = 3, color =
"#7fa900" });
 ViewBag.Conferences = conferences;
 string[] resources = new string[] { "Conferences" };
 ViewBag.Resources = resources;
 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 13, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 13, 12, 30, 0),
 IsAllDay = false,
 ConferenceId = new int[] { 1, 2 }
 });
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Testing",
 StartTime = new DateTime(2023, 2, 14, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 12, 30, 0),
 IsAllDay = false,
 ConferenceId = new int[] { 1, 2, 3 }
 });
 return appData;
 }
 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int[] ConferenceId { get; set; }
 }
 public class ConferenceRes
 {
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 }

```

### Simple resource header customization

It is possible to customize the resource header cells using built-in template option and change the look and appearance of it in both the vertical and timeline view modes. All the resource related fields and other information can be accessed within the resource header template option.

**Example:** To customize the resource header and display it along with designation field, refer the below code example.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .ResourceHeaderTemplate("#resourceTemplate")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res =>
 {
 res.DataSource(ViewBag.Doctors).Field("DoctorId").Title("Doctor
Name").Name("Doctors").TextField("text").IdField("id").ColorField("color").A
dd();
 })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)
<script id="resourceTemplate" type="text/x-template">
 <div class='template-wrap'>
 <div class="resource-image"></div>
 <div class="resource-details">
 <div class="resource-name">${getDoctorName(data)}</div>
 <div class="resource-designation">${getDoctorLevel(data)}</div>
 </div>
 </div>
</script>
<script type="text/javascript">
 window.getDoctorName = function (value) {
 return (value.resourceData) ?
value.resourceData[value.resource.textField] : value.resourceName;
 };
 window.getDoctorImage = function (value) {
 var resourceName = window.getDoctorName(value);
 return resourceName.replace(' ', '-').toLowerCase();
 };
 window.getDoctorLevel = function (value) {
 var resourceName = window.getDoctorName(value);
 return (resourceName === 'Nancy') ? 'Cardiologist' : (resourceName
=== 'Alice') ? 'Neurologist' : 'Orthopedic Surgeon';
 };
</script>
<style>
 .e-schedule .e-vertical-view .e-resource-cells {
 height: 62px;
 }
 .e-schedule .template-wrap {
 display: flex;
 text-align: left;
 }

 .e-schedule .template-wrap .resource-image img {
```

```

 width: 45px;
 height: 45px;
 }
 .e-schedule .template-wrap .resource-details {
 padding-left: 10px;
 }
 .e-schedule .template-wrap .resource-details .resource-name {
 font-size: 16px;
 font-weight: 500;
 margin-top: 5px;
 }
 .e-schedule.e-device .template-wrap .resource-details .resource-name {
 font-size: inherit;
 font-weight: inherit;
 }
 .e-schedule.e-device .e-resource-tree-popup .e-fullrow {
 height: 50px;
 }
 .e-schedule.e-device .template-wrap .resource-details .resource-
designation {
 display: none;
 }
</style>

```

**DATA.CS**

```

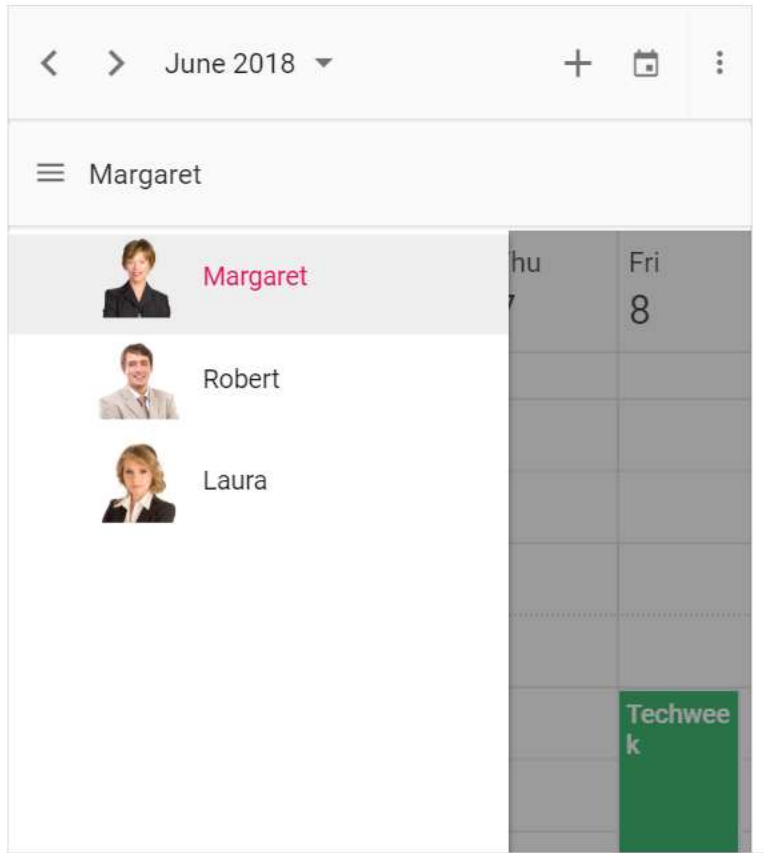
public ActionResult Index()
{
 List<DoctorResources> doctors = new List<DoctorResources>();
 doctors.Add(new DoctorResources { text = "Nancy", id = 1, color =
"#ea7a57" });
 doctors.Add(new DoctorResources { text = "Alice", id = 2, color =
"#7fa900" });
 doctors.Add(new DoctorResources { text = "Robson", id = 3, color =
"#7499e1" });
 ViewBag.Doctors = doctors;
 string[] resources = new string[] { "Doctors" };
 ViewBag.Resources = resources;
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineWeek },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineMonth },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Agenda }
 };
 ViewBag.view = viewOption;
 return View();
}
public class DoctorResources
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
}

```

```
}

```

**Note:** To customize the resource header in compact mode properly make use of the class `e-device` as in the code example.



#### Customizing resource header with multiple columns

It is possible to customize the resource headers to display with multiple columns such as Room, Type and Capacity. The following code example depicts the way to achieve it and is applicable only on timeline views.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .ResourceHeaderTemplate("#resourceTemplate")
 .Group(group => group.Resources(ViewBag.ResourceNames))
 .Resources(res =>
 {
 res.AllowMultiple(true).DataSource(ViewBag.RoomDatas).Field("RoomId").Title("Room
 Type").Name("MeetingRoom").TextField("name").IdField("id").ColorField("color")
 .Add();
 })
 .Views(view =>
```

```

 {
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .RenderCell("onRenderCell")
 .SelectedDate(new DateTime(2018, 7, 30))
 .Render()
)
<script id="resourceTemplate" type="text/x-template">
<div class='template-wrap'>
 <div class="room-name">${getRoomName(data)}</div>
 <div class="room-type">${getRoomType(data)}</div>
 <div class="room-capacity">${getRoomCapacity(data)}</div>
</div>
</script>
<script type="text/javascript">
 window.getRoomName = function (value) {
 return value.resourceData[value.resource.textField];
 };
 window.getRoomType = function (value) {
 return value.resourceData.type;
 };
 window.getRoomCapacity = function (value) {
 return value.resourceData.capacity;
 };
 function onRenderCell(args) {
 if (args.elementType === 'emptyCells' &&
args.element.classList.contains('e-resource-left-td')) {
 var target = args.element.querySelector('.e-resource-text');
 target.innerHTML = '<div class="name">Rooms</div><div
class="type">Type</div><div class="capacity">Capacity</div>';
 }
 }
</script>
<style>
 .e-schedule .e-timeline-view .e-resource-left-td,
 .e-schedule .e-timeline-month-view .e-resource-left-td {
 vertical-align: bottom;
 }
 .e-schedule.e-device .e-timeline-view .e-resource-left-td,
 .e-schedule.e-device .e-timeline-month-view .e-resource-left-td {
 width: 75px;
 }
 .e-schedule .e-timeline-view .e-resource-left-td .e-resource-text,
 .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-text
{
 display: flex;
 font-weight: 500;
 padding: 0;
 }
 .e-schedule .e-timeline-view .e-resource-left-td .e-resource-text>div,
 .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div {
 border-right: 1px solid rgba(0, 0, 0, 0.12);
 border-top: 1px solid rgba(0, 0, 0, 0.12);
 flex: 0 0 33.3%;
 }

```

```

 font-weight: 500;
 height: 36px;
 line-height: 34px;
 padding-left: 5px;
 }
 .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div {
 border-top: 0;
 }
 .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div:last-child,
 .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div:last-child {
 border-right: 0;
 }
 .e-schedule .template-wrap {
 display: flex;
 height: 100%;
 text-align: left;
 }
 .e-schedule .template-wrap>div {
 border-right: 1px solid rgba(0, 0, 0, 0.12);
 flex: 0 0 33.3%;
 font-weight: 500;
 line-height: 58px;
 overflow: hidden;
 padding-left: 5px;
 text-overflow: ellipsis;
 }
 .e-schedule .template-wrap>div:last-child {
 border-right: 0;
 }
 .e-schedule .e-timeline-view .e-resource-cells,
 .e-schedule .e-timeline-month-view .e-resource-cells {
 padding-left: 0;
 }
 .e-schedule .e-timeline-view .e-date-header-wrap table col,
 .e-schedule .e-timeline-view .e-content-wrap table col {
 width: 100px;
 }
 @@media (max-width: 550px) {
 .e-schedule .e-timeline-view .e-resource-left-td,
 .e-schedule .e-timeline-month-view .e-resource-left-td {
 width: 100px;
 }
 .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div,
 .e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div,
 .e-schedule .template-wrap>div {
 flex: 0 0 100%;
 }
 .e-schedule .template-wrap>div:first-child {
 border-right: 0;
 }
 .e-schedule .e-timeline-view .e-resource-left-td .e-resource-
text>div:first-child,

```



```

.e-schedule .e-timeline-month-view .e-resource-left-td .e-resource-
text>div:first-child {
 border-right: 0;
}
.e-schedule .room-type,
.e-schedule .room-capacity {
 display: none;
}
}
</style>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetRoomData();
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { name = "Jammy", id = 1, color = "#ea7a57",
 capacity = 20, type = "Conference" });
 rooms.Add(new RoomResource { name = "Tweety", id = 2, color = "#7fa900",
 capacity = 7, type = "Cabin" });
 rooms.Add(new RoomResource { name = "Nestle", id = 3, color = "#5978ee",
 capacity = 5, type = "Cabin" });
 rooms.Add(new RoomResource { name = "Phoenix", id = 4, color =
 "#fec200", capacity = 15, type = "Conference" });
 rooms.Add(new RoomResource { name = "Mission", id = 5, color =
 "#df5286", capacity = 25, type = "Conference" });
 rooms.Add(new RoomResource { name = "Hangout", id = 6, color =
 "#00bdae", capacity = 10, type = "Cabin" });
 rooms.Add(new RoomResource { name = "Rick Roll", id = 7, color =
 "#865fcf", capacity = 20, type = "Conference" });
 rooms.Add(new RoomResource { name = "Rainbow", id = 8, color =
 "#1aaa55", capacity = 8, type = "Cabin" });
 rooms.Add(new RoomResource { name = "Swarm", id = 9, color = "#df5286",
 capacity = 30, type = "Conference" });
 rooms.Add(new RoomResource { name = "Photogenic", id = 10, color =
 "#710193", capacity = 25, type = "Conference" });
 ViewBag.RoomDatas = rooms;
 string[] resources = new string[] { "MeetingRoom" };
 ViewBag.ResourceNames = resources;
 return View();
}

public class RoomResource
{
 public int id { set; get; }
 public string name { set; get; }
 public string color { set; get; }
 public int capacity { set; get; }
 public string type { set; get; }
}

public List<RoomData> GetRoomData()
{
 List<RoomData> roomData = new List<RoomData>();
 roomData.Add(new RoomData
 {
 Id = 1,

```

```

 Subject = "Board Meeting",
 Description = "Meeting to discuss business goal of 2018.",
 StartTime = new DateTime(2023, 7, 30, 9, 0, 0),
 EndTime = new DateTime(2023, 7, 30, 11, 0, 0),
 RoomId = 1
 });
 roomData.Add(new RoomData
 {
 Id = 2,
 Subject = "Training session on JSP",
 Description = "Knowledge sharing on JSP topics.",
 StartTime = new DateTime(2023, 7, 30, 15, 0, 0),
 EndTime = new DateTime(2023, 7, 30, 17, 0, 0),
 RoomId = 3
 });
 roomData.Add(new RoomData
 {
 Id = 3,
 Subject = "Sprint Planning with Team members",
 Description = "Planning tasks for sprint.",
 StartTime = new DateTime(2023, 7, 30, 9, 30, 0),
 EndTime = new DateTime(2023, 7, 30, 11, 0, 0),
 RoomId = 5
 });
 return roomData;
}
public class RoomData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Description { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public int RoomId { get; set; }
}

```

### Collapse/Expand child resources in timeline views

It is possible to expand and collapse the resources which have child resource in timeline views dynamically. By default, resources are in expanded state with their child resource. We can collapse and expand the child resources in UI by setting `ExpandedField` option as `false` whereas its default value is `true`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

 res.AllowMultiple(false).DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").ColorField("RoomColor").ExpandedField("IsExpand").Add();
 }

```

```

res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2023, 4, 1))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasources = GetResourceData();
 // datasource for room resources
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { RoomText = "ROOM 1", Id = 1, RoomColor = "#cb6bb2", IsExpand = false });
 rooms.Add(new RoomResource { RoomText = "ROOM 2", Id = 2, RoomColor = "#56ca85", IsExpand = true });
 ViewBag.Rooms = rooms;
 // datasource for owner resources
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor = "#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor = "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3, OwnerGroupId = 1, OwnerColor = "#7499e1" });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Rooms", "Owners" };
 return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 1,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",

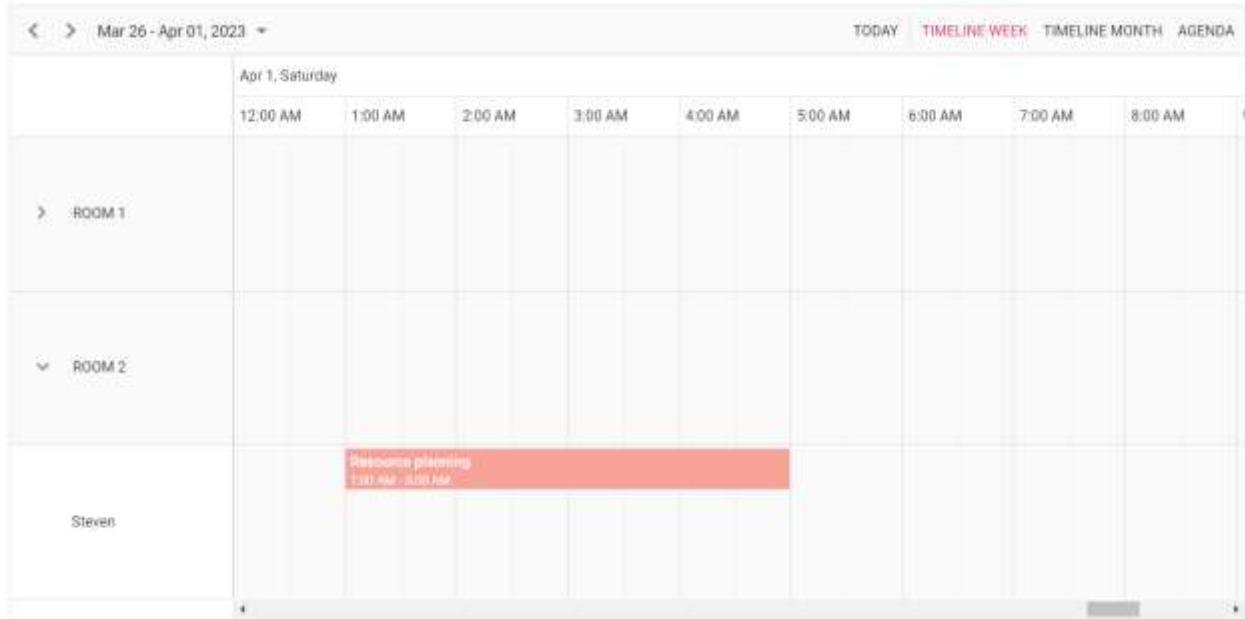
```

```
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 3,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2023, 4, 1, 1, 0, 0),
 EndTime = new DateTime(2023, 4, 1, 5, 0, 0),
 IsAllDay = false,
 OwnerId = 2,
 RoomId = 2
 });
 return resourceData;
}

public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
 public int RoomId { get; set; }
}

public class RoomResource
{
 public string RoomText { set; get; }
 public int Id { set; get; }
 public string RoomColor { set; get; }
 public bool IsExpand { get; set; }
}

public class OwnerResource
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
 public int OwnerGroupId { set; get; }
}
```



### Displaying tooltip for resource headers

It is possible to display tooltip over the resource headers showing the resource information. By default, there won't be any tooltip displayed on the resource headers, and to enable it, you need to assign the customized template design to the `HeaderTooltipTemplate` option within the `Group` property.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group =>
group.Resources(ViewBag.Resources).HeaderTooltipTemplate("#tooltipTemplate")
)
 .Resources(res => {
res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.Week).Add();
 view.Option(View.Month).Add();
 view.Option(View.Agenda).Add();
 })
 .Render()
)
<script id="tooltipTemplate" type="text/x-template">
 <div class='template-wrap'>
 <div class="res-text">Name: ${resourceData.OwnerText} </div>
 </div>
</script>
<style>
 .e-schedule .e-vertical-view .e-resource-cells {
 height: 45px;
```

```

 }

 .e-schedule .e-agenda-view .template-wrap .resource-text {
 text-align: center;
 }
 .e-schedule .template-wrap .resource-text {
 font-size: 15px;
 padding: 4px 4px 4px;
 height: 25px;
 text-overflow: ellipsis;
 white-space: nowrap;
 overflow: hidden;
 }
</style>

```

### DATA.CS

```

public ActionResult Index()
{
 List<OwnerResources> owners = new List<OwnerResources>();
 owners.Add(new OwnerResources { OwnerText = "Nancy", Id = 1, OwnerColor = "#ffaa00" });
 owners.Add(new OwnerResources { OwnerText = "Steven", Id = 2, OwnerColor = "#f8a398" });
 owners.Add(new OwnerResources { OwnerText = "Michael", Id = 3, OwnerColor = "#7499e1" });
 ViewBag.Owners = owners;
 string[] resources = new string[] { "Owners" };
 ViewBag.Resources = resources;
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineMonth },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Agenda }
 };
 ViewBag.view = viewOption;
 return View();
}

public class OwnerResources
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
}

```

### Choosing between resource colors for appointments

By default, the colors defined on the top level resources collection will be applied for the events. In case, if you want to apply specific resource color to events irrespective of its top-level parent resource color, it can be achieved by defining `ResourceColorField` option within the `EventSettings` property.

In the following example, the colors mentioned in the second level will get applied over the events.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@Html.EJS().RadioButton("radio1").Label("Rooms").Name("default").Value("Rooms").Checked(true).Change("onChange").Render()
@Html.EJS().RadioButton("radio2").Label("Owners").Name("default").Value("Owners").Checked(false).Change("onChange").Render()
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res =>
 {

res.DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").GroupIDField("RoomGroupId").ColorField("RoomColor").AllowMultiple(false).Add();

res.DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").AllowMultiple(true).Add();
 })
 .EventSettings(e =>
e.DataSource(ViewBag.datasource).ResourceColorField("Rooms")
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 4, 3))
 .Render()
)
<script type="text/javascript">
 function onChange(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.eventSettings.resourceColorField = args.value;
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetResourceTeamData();
 // datasource for Room resources
 List<RoomResources> rooms = new List<RoomResources>();
 rooms.Add(new RoomResources { RoomText = "ROOM 1", Id = 1, RoomGroupId = 1, RoomColor = "#cb6bb2" });
 rooms.Add(new RoomResources { RoomText = "ROOM 2", Id = 2, RoomGroupId = 2, RoomColor = "#56ca85" });
 ViewBag.Rooms = rooms;
 // datasource for Owner resources
 List<OwnerResources> owners = new List<OwnerResources>();
 owners.Add(new OwnerResources { OwnerText = "Nancy", Id = 1, OwnerGroupId = 1, OwnerColor = "#1aaa55" });
 owners.Add(new OwnerResources { OwnerText = "Steven", Id = 2, OwnerGroupId = 2, OwnerColor = "#7fa900" });
 ViewBag.Owners = owners;
 // Scheduler resource names

```

```

ViewBag.Resources = new string[] { "Rooms", "Owners" };
// Scheduler views
List<ScheduleView> viewOption = new List<ScheduleView>()
{
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineWeek },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineMonth },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Agenda }
};
ViewBag.view = viewOption;
return View();
}
public List<ResourceData> GetResourceTeamData()
{
 List<ResourceData> resourceTeamData = new List<ResourceData>();
 resourceTeamData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Decoding",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 11, 30, 0),
 IsAllDay = false,
 RoomId = 1,
 OwnerId = 1
 });
 resourceTeamData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Bug Automation",
 StartTime = new DateTime(2023, 4, 4, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 4, 11, 30, 0),
 IsAllDay = false,
 RoomId = 2,
 OwnerId = 2
 });
 return resourceTeamData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int RoomId { get; set; }
 public int OwnerId { get; set; }
}
public class RoomResources
{
 public string RoomText { set; get; }
 public int Id { set; get; }
 public int RoomGroupId { set; get; }
 public string RoomColor { set; get; }
}

```



```
public class OwnerResources
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public int OwnerGroupId { set; get; }
 public string OwnerColor { set; get; }
}
```

**Note:** The value of the `ResourceColorField` field should be mapped with the `Name` value given within the `Resources` property.

#### Dynamically add and remove resources

It is possible to add or remove the resources dynamically to and from the Scheduler respectively. In the following example, when the checkbox is checked and unchecked, the respective resources gets added up or removed from the Scheduler layout. To add new resource dynamically, `addResource` method is used which accepts the arguments such as resource object, resource name (within which level, the resource object to be added) and index (position where the resource needs to be added).

To remove the resources dynamically, `removeResource` method is used which accepts the index (position from where the resource to be removed) and resource name (within which level, the resource object presents) as parameters.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@Html.EJS().CheckBox("company").Change("onChange").Checked(true).Disabled(true).Label("Company").Value("1").Render()
@Html.EJS().CheckBox("birthdays").Change("onChange").Checked(false).Label("Birthday").Value("2").Render()
@Html.EJS().CheckBox("holidays").Change("onChange").Checked(false).Label("Holiday").Value("3").Render()
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

 res.DataSource(ViewBag.Calendars).Field("CalendarId").Title("Calendars").Name("Calendars").TextField("CalendarName").IdField("CalendarId").ColorField("CalendarColor").AllowMultiple(true).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 3, 1))
 .Render()
)
<style>
 .property-panel-content .e-checkbox-wrapper.company .e-frame {
 background-color: #ff7f50;
 border-color: transparent;
 }
 .property-panel-content .e-checkbox-wrapper.birthday .e-frame {
 background-color: #AF27CD;
 border-color: transparent;
 }
}
```

```

 .property-panel-content .e-checkbox-wrapper.holiday .e-frame {
 background-color: #808000;
 border-color: transparent;
 }
 .e-schedule .e-month-view .e-appointment {
 border-color: transparent;
 }
 .highcontrast .property-panel-content .e-checkbox-wrapper .e-frame.e-
check,
 .bootstrap .property-panel-content .e-checkbox-wrapper .e-frame.e-check
 {
 color: #fff;
 }
</style>
<script type="text/javascript">
 function onChange(args) {
 var calendarCollections = [
 { CalendarName: 'Company', CalendarId: 1, CalendarColor:
'#ff7f50' },
 { CalendarName: 'Birthday', CalendarId: 2, CalendarColor:
'#AF27CD' },
 { CalendarName: 'Holiday', CalendarId: 3, CalendarColor:
'#808000' }
];
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var value = parseInt(args.event.target.getAttribute('value'), 10);
 var resourceData = calendarCollections.filter(function (calendar) {
return calendar.CalendarId === value; });
 if (args.checked) {
 scheduleObj.addResource(resourceData[0], 'Calendars', value -
1);
 } else {
 scheduleObj.removeResource(value, 'Calendars');
 }
 }
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 List<CalendarRes> calendarCollections = new List<CalendarRes>();
 calendarCollections.Add(new CalendarRes { CalendarName = "Company",
CalendarId = 1, CalendarColor = "#c43081" });
 ViewBag.Calendars = calendarCollections;
 // scheduler views
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 // Scheduler default resource name
 string[] resources = new string[] { "Calendars" };
 ViewBag.Resources = resources;
 // Scheduler datasource
}

```

```

 ViewBag.datasource = GetCalendarData();
 return View();
 }
 public class CalendarRes
 {
 public string CalendarName { set; get; }
 public int CalendarId { set; get; }
 public string CalendarColor { set; get; }
 }
 public List<ResourceEventsData> GetCalendarData()
 {
 List<ResourceEventsData> calendarData = new List<ResourceEventsData>();
 calendarData.Add(new ResourceEventsData
 {
 Id = 1,
 Subject = "Conference meeting",
 StartTime = new DateTime(2018, 3, 1),
 EndTime = new DateTime(2018, 3, 2),
 IsAllDay = true,
 CalendarId = 1
 });
 calendarData.Add(new ResourceEventsData
 {
 Id = 2,
 Subject = "Gladys Spellman",
 StartTime = new DateTime(2018, 3, 8),
 EndTime = new DateTime(2018, 3, 9),
 IsAllDay = true,
 CalendarId = 2
 });
 calendarData.Add(new ResourceEventsData
 {
 Id = 3,
 Subject = "Global Family Day",
 StartTime = new DateTime(2018, 3, 15),
 EndTime = new DateTime(2018, 3, 16),
 IsAllDay = true,
 CalendarId = 3
 });
 return calendarData;
 }
 public class ResourceEventsData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int CalendarId { get; set; }
 }
}

```

### Setting different working days and hours for resources

Each resource in the Scheduler can have different working hours as well as different working days set to it. There are default options available within the **Resources** collection, to customize the default working hours and days of the Scheduler.

*Set different work days*

Different working days can be set for the resources of Scheduler using the **WorkDaysField** property which maps the working days field from the resource dataSource. This field accepts the collection of day indexes (from 0 to 6) of a week. By default, it is set to [1, 2, 3, 4, 5] and in the following example, each resource has been set with different values and therefore each of them will render only those working days. This option is applicable only on the calendar views and is not applicable on timeline views.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {
 res.DataSource(ViewBag.Doctors).Field("DoctorId").Title("Doctor
Name").Name("Doctors").TextField("text").IdField("id").ColorField("color").W
orkDaysField("workDays").Add(); }).Views(view => {
view.Option(View.WorkWeek).Add(); view.Option(View.Month).Add();
}))
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasources = GetDoctorData();
 List<DoctorRes> doctors = new List<DoctorRes>();
 doctors.Add(new DoctorRes { text = "Will Smith", id = 1, color =
"#ea7a57", workDays = new List<int> { 1, 2, 4, 5 } });
 doctors.Add(new DoctorRes { text = "Alice", id = 2, color = "rgb(53,
124, 210)", workDays = new List<int> { 1, 3, 5 } });
 doctors.Add(new DoctorRes { text = "Robson", id = 3, color =
"#7fa900" });
 ViewBag.Doctors = doctors;
 string[] resources = new string[] { "Doctors" };
 ViewBag.Resources = resources;
 return View();
}

public class DoctorRes
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 public List<int> workDays { set; get; }
 public string startHour { set; get; }
 public string endHour { set; get; }
}

public List<DoctorData> GetDoctorData()
{
 List<DoctorData> doctorData = new List<DoctorData>();
 doctorData.Add(new DoctorData
 {
```

```

 Id = 1,
 Subject = "Echocardiogram",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 11, 30, 0),
 IsAllDay = false,
 DoctorId = 1
 });
 doctorData.Add(new DoctorData
 {
 Id = 2,
 Subject = "Lumbar punctures",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 10, 45, 0),
 IsAllDay = false,
 DoctorId = 2
 });
 doctorData.Add(new DoctorData
 {
 Id = 3,
 Subject = "Osteoarthritis",
 StartTime = new DateTime(2023, 4, 4, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 4, 12, 0, 0),
 IsAllDay = false,
 DoctorId = 3
 });
 return doctorData;
}
public class DoctorData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int DoctorId { get; set; }
}

```

### Set different work hours

Working hours indicates the work hour duration of a day, which is highlighted visually with active color over the work cells. Each resource on the Scheduler can be defined with its own set of working hours as depicted in the following example.

- **StartHourField** - Denotes the start time of the working/business hour in a day.
- **EndHourField** - Denotes the end time limit of the working/business hour in a day.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

```

```

 res.DataSource(ViewBag.Doctors).Field("DoctorId").Title("Doctor
Name").Name("Doctors").TextField("text").IdField("id").ColorField("color").S
tartHourField("startHour").EndHourField("endHour").Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 4, 1))
 .Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasources = GetDoctorData();
 List<DoctorRes> doctors = new List<DoctorRes>();
 doctors.Add(new DoctorRes { text = "Will Smith", id = 1, color =
"#ea7a57", startHour = "08:00", endHour = "15:00" });
 doctors.Add(new DoctorRes { text = "Alice", id = 2, color = "rgb(53,
124, 210)", startHour = "09:00", endHour = "17:00" });
 doctors.Add(new DoctorRes { text = "Robson", id = 3, color = "#7fa900",
startHour = "08:00", endHour = "16:00" });
 ViewBag.Doctors = doctors;
 // Scheduler views
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineWeek
 },
 new ScheduleView { Option =
Syncfusion.EJ2.Schedule.View.TimelineMonth }
 };
 ViewBag.view = viewOption;
 string[] resources = new string[] { "Doctors" };
 ViewBag.Resources = resources;
 return View();
}

public class DoctorRes
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 public string startHour { set; get; }
 public string endHour { set; get; }
}

public List<DoctorData> GetDoctorData()
{
 List<DoctorData> doctorData = new List<DoctorData>();
 doctorData.Add(new DoctorData
 {
 Id = 1,
 Subject = "Echocardiogram",
 StartTime = new DateTime(2023, 4, 2, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 2, 11, 30, 0),
 IsAllDay = false,
 }

```

```

 DoctorId = 1
 });
 doctorData.Add(new DoctorData
 {
 Id = 2,
 Subject = "Lumbar punctures",
 StartTime = new DateTime(2023, 4, 2, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 2, 10, 45, 0),
 IsAllDay = false,
 DoctorId = 2
 });
 doctorData.Add(new DoctorData
 {
 Id = 3,
 Subject = "Osteoarthritis",
 StartTime = new DateTime(2023, 4, 4, 10, 0, 0),
 EndTime = new DateTime(2023, 4, 4, 12, 0, 0),
 IsAllDay = false,
 DoctorId = 3
 });
 return doctorData;
}
public class DoctorData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int DoctorId { get; set; }
}

```

In this example, a resource named **Will Smith** is depicted with working hours ranging from 8.00 AM to 3.00 PM and is visually illustrated with active colors, whereas the other two resources have different working hours set.

#### Hide non-working days when grouped by date

In Scheduler, you can set custom work days for each resource and group the Scheduler by date to display these work days. By default, the Scheduler will show all days when it is grouped by date, even if they are not included in the custom work days for the resources. However, you can use the **HideNonWorkingDays** property to only display the custom work days in the Scheduler.

To use the **HideNonWorkingDays** property, you need to include it in the configuration options for your Scheduler component. Set the value of **HideNonWorkingDays** to **true** to enable this feature.

**Example:** To display the Scheduler with resources grouped by date for custom working days,

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Group(group =>
 group.ByDate(true).HideNonWorkingDays(true).Resources(ViewBag.Resources))

```

```

 .Resources(res => {

res.DataSource(ViewBag.Owners).Field("OwnerId").Title("Assignee").Name("Owners").TextField("text").IdField("id").ColorField("color").AllowMultiple(true).WorkDaysField("workDays").Add();
 })
 .Views(view => {
 view.Option(View.Week).Add();
 view.Option(View.Month).Add();
 view.Option(View.Agenda).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2023, 18, 1))
 .Render()
)
}

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasources = GetResourceData();
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { text = "Alice", id = 1, color = "#ffaa00", workDays: [1, 2, 3, 4] });
 owners.Add(new OwnerResource { text = "Smith", id = 2, color = "#f8a398", workDays: [2, 3, 5] });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Owners" };
 return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2022, 18, 3, 10, 0, 0),
 EndTime = new DateTime(2022, 18, 3, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2022, 18, 4, 10, 0, 0),
 EndTime = new DateTime(2022, 18, 4, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 2
 });
 return resourceData;
}

public class ResourceData
{
 public int Id { get; set; }
}

```



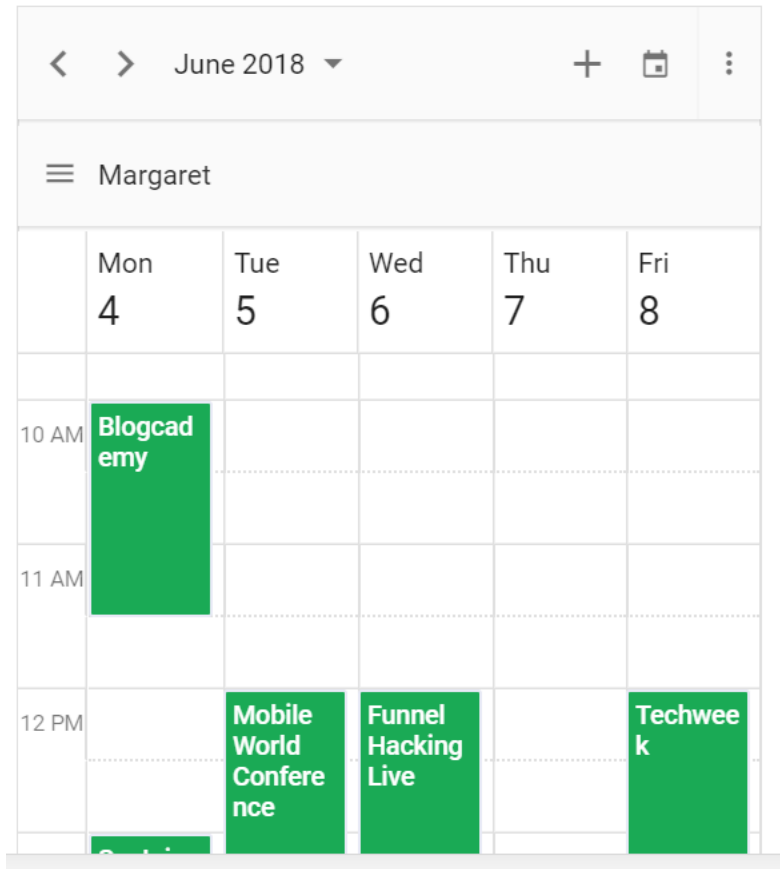
```
public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
public int OwnerId { get; set; }
}
public class OwnerResource
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 public int[] workDays { set; get; }
}
```

**Note:** The `HideNonWorkingDays` property only applies when the Scheduler is grouped `byDate`.

#### Compact view in mobile

Although the Scheduler views are designed keeping in mind the responsiveness of the control in mobile devices, however when using Scheduler with multiple resources - it is difficult to view all the resources and its relevant events at once on the mobile. Therefore, we have introduced a new compact mode specially for displaying multiple resources of Scheduler on mobile devices. By default, this mode is enabled while using Scheduler with multiple resources on mobile devices. If in case, you need to disable this compact mode, set `false` to the `EnableCompactView` option within the `Group` property. Disabling this option will display the exact desktop mode of Scheduler view on mobile devices.

With this compact view enabled on mobile, you can view only single resource at a time and to switch to other resources, there is a TreeView at the left listing out all other available resources - clicking on which will display that particular resource and its related appointments.



### Adaptive UI in desktop

By default, the Scheduler layout adapts automatically in the desktop and mobile devices with appropriate UI changes. In case, if the user wants to display the Adaptive scheduler in desktop mode with adaptive enhancements, then the property `EnableAdaptiveUI` can be set to true. Enabling this option will display the exact mobile mode of Scheduler view on desktop devices.

Some of the default changes made for compact Scheduler to render in desktop devices are as follows,

- View options displayed in the Navigation drawer.
- Plus icon is added to the header for new event creation.
- Today icon is added to the header instead of the Today button.
- With Multiple resources – only one resource has been shown to enhance the view experience of resource events details clearly. To switch to other resources, there is a TreeView on the left that lists all other available resources, clicking on which will display that particular resource and its related events.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@Html.EJS()
 .Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .EnableAdaptiveUI(true)
 .CurrentView(View.Month)
```

```

.Views(view =>
{
view.Option(View.Day).Add();
view.Option(View.Week).Add();
view.Option(View.Month).Add();
}).Group(group => group.Resources(ViewBag.Resources)).Resources(res =>
{
res.DataSource(ViewBag.Projects).Field("ProjectId").Title("Choose
Project").Name("Projects").TextField("text").IdField("id").ColorField("color
").Add();

res.DataSource(ViewBag.Categories).Field("TaskId").Title("Category").Name("C
ategories").TextField("text").IdField("id").GroupIDField("groupId").ColorFie
ld("color").AllowMultiple(true).Add();
})
.EventSettings(e => e.Fields(f => f.Subject(sub =>
sub.Name("Subject").Title("Summary")).Description(des =>
des.Name("Description").Title("Comments"))).DataSource(ViewBag.datasources)
.SelectedDate(new DateTime(2018, 4, 4))
.Render()

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasources = GetTimelineResourceData();
 // datasource for project resources
 List<ResourceDataSourceModel> projects = new
List<ResourceDataSourceModel>();
 projects.Add(new ResourceDataSourceModel { text = "PROJECT 1", id = 1,
color = "#cb6bb2" });
 projects.Add(new ResourceDataSourceModel { text = "PROJECT 2", id = 2,
color = "#56ca85" });
 projects.Add(new ResourceDataSourceModel { text = "PROJECT 3", id = 3,
color = "#df5286" });
 ViewBag.Projects = projects;
 // datasource for category resources
 List<ResourceDataSourceModel> categories = new
List<ResourceDataSourceModel>();
 categories.Add(new ResourceDataSourceModel { text = "Nancy", id = 1,
groupId = 1, color = "#df5286" });
 categories.Add(new ResourceDataSourceModel { text = "Steven", id = 2,
groupId = 1, color = "#7fa900" });
 categories.Add(new ResourceDataSourceModel { text = "Robert", id = 3,
groupId = 2, color = "#ea7a57" });
 categories.Add(new ResourceDataSourceModel { text = "Smith", id = 4,
groupId = 2, color = "#5978ee" });
 categories.Add(new ResourceDataSourceModel { text = "Michael", id = 5,
groupId = 3, color = "#df5286" });
 categories.Add(new ResourceDataSourceModel { text = "Root", id = 6,
groupId = 3, color = "#00bdae" });
 ViewBag.Categories = categories;
 ViewBag.Resources = new string[] { "Projects", "Categories" };
 return View();
}

public List<ResourceData> GetTimelineResourceData()

```

```
{
 List<ResourceData> timelineResourceData = new List<ResourceData>();
 timelineResourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Decoding",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 1,
 TaskId = 1
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Bug Automation",
 StartTime = new DateTime(2023, 4, 4, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 4, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 2,
 TaskId = 1
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Functionality testing",
 StartTime = new DateTime(2023, 4, 5, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 5, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 3,
 TaskId = 1
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 4,
 Subject = "Resolution-based testing",
 StartTime = new DateTime(2023, 4, 3, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 3, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 1,
 TaskId = 2
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 5,
 Subject = "Test report Validation",
 StartTime = new DateTime(2023, 4, 4, 9, 30, 0),
 EndTime = new DateTime(2023, 4, 4, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 2,
 TaskId = 2
 });
 timelineResourceData.Add(new ResourceData
 {
 Id = 6,
 Subject = "Test case correction",
 StartTime = new DateTime(2023, 4, 5, 9, 30, 0),
```

```

 EndTime = new DateTime(2023, 4, 5, 11, 30, 0),
 IsAllDay = false,
 ProjectId = 3,
 TaskId = 2
 });
 return timelineResourceData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int ProjectId { get; set; }
 public int TaskId { get; set; }
}
public class ResourceDataSourceModel
{
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 public int groupId { set; get; }
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Timeline header rows

The Timeline views can have additional header rows other than its default date and time header rows. It is possible to show individual header rows for displaying year, month and week separately using the `HeaderRows` property. This property is applicable only on the timeline views. The possible rows which can be added using `HeaderRows` property are as follows.

- Year
- Month
- Week
- Date
- Hour

**Note:** The `Hour` row is not applicable for Timeline month view.

The following example shows the Scheduler displaying all the available header rows on timeline views.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .HeaderRows(headerRow => {
 headerRow.Option(HeaderRowType.Year).Add();
 }

```

```

 headerRow.Option(HeaderRowType.Month).Add();
 headerRow.Option(HeaderRowType.Week).Add();
 headerRow.Option(HeaderRowType.Date).Add();
 headerRow.Option(HeaderRowType.Hour).Add();
 })
 .Views(view => { view.Option(View.TimelineWeek).Add(); })
 .SelectedDate(new DateTime(2018, 1, 1))
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 return View();
}

```

**Display year and month rows in timeline views**

To display the timeline Scheduler simply with year and month names alone, define the option **Year** and **Month** within the **HeaderRows** property.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .HeaderRows(headerRow => {
 headerRow.Option(HeaderRowType.Year).Add();
 headerRow.Option(HeaderRowType.Month).Add();
 })
 .Views(view => {
 view.Option(View.TimelineMonth).Add();
 })
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 return View();
}

```

**Display week numbers in timeline views**

The week number can be displayed in a separate header row of the timeline Scheduler by setting **Week** option within **HeaderRows** property.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")

```

```

 .HeaderRows(headerRow => {
 headerRow.Option(HeaderRowType.Week).Add();
 headerRow.Option(HeaderRowType.Date).Add();
 headerRow.Option(HeaderRowType.Hour).Add();
 })
 .Views(view => {
 view.Option(View.TimelineMonth).Interval(24).Add();
 view.Option(View.TimelineWeek).Interval(3).Add();
 view.Option(View.TimelineDay).Interval(4).Add();
 })
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 return View();
}

```

**Timeline view displaying dates of a complete year**

It is possible to display a complete year in a timeline view by setting **Interval** value as 12 and defining **TimelineMonth** view option within the **Views** property of Scheduler.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .HeaderRows(headerRow => {
 headerRow.Option(HeaderRowType.Month).Add();
 headerRow.Option(HeaderRowType.Date).Add();
 })
 .Views(view => {
 view.Option(View.TimelineMonth).Interval(12).Add();
 })
 .Render()
)

```

**DATA.CS**

```

public ActionResult Index()
{
 return View();
}

```

**Customizing the header rows using template**

You can customize the text of the header rows and display any images or formatted text on each individual header rows using the built-in **Template** option available within the **HeaderRows** property.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule

```

```
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .HeaderRows(headerRow => {
 headerRow.Option(HeaderRowType.Month).Template("#month-
template").Add();
 headerRow.Option(HeaderRowType.Week).Template("#week-
template").Add();
 headerRow.Option(HeaderRowType.Date).Add();
 })
 .Views(view => {
 view.Option(View.TimelineDay).Add();
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineWorkWeek).Add();
 view.Option(View.TimelineMonth).Add();
 })
 .Render()
)
<script id="month-template" type="text/x-template">
 ${getMonthDetails(data)}
</script>
<script id="week-template" type="text/x-template">
 ${getWeekDetails(data)}
</script>
<script type="text/javascript">
 var instance = new ej.base.Internationalization();
 window.getMonthDetails = function (value) {
 return instance.formatDate(value.date, { skeleton: 'yMMMM' });
 };
 window.getWeekDetails = function (value) {
 return 'Week ' + ej.schedule.getWeekNumber(value.date);
 };
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Row Auto Height

By default, the height of the Scheduler rows in Timeline views are static and therefore, when the same time range holds multiple overlapping appointments, a **+n more** text indicator will be displayed. With this feature enabled, you can now view all the overlapping appointments present in those specific time range by auto-adjusting the row height based on the presence of the appointments count, instead of displaying the **+n more** text indicators.

To enable auto row height adjustments on Scheduler Timeline views and Month view, set **true** to the **rowAutoHeight** property whose default value is **false**.



**Note:** This auto row height adjustment is applicable only on all the Timeline views as well as on the calendar Month view.

Now, let's see how it works on those applicable views with examples.

### Calendar month view

By default, the rows of the calendar Month view can hold only the limited appointments count based on its row height, and the rest of the overlapping appointments are indicated with a +n more text indicator. The following example shows how the month view row auto-adjusts based on the number of appointments count, when this RowAutoHeight feature is enabled.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Month).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .RowAutoHeight(true)
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2023, 1, 10, 12, 0, 0), EndTime = new
 DateTime(2023, 1, 10, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2023, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2023, 1, 10, 13, 0, 0), EndTime = new
 DateTime(2023, 1, 10, 14, 30, 0) });
 appData.Add(new AppointmentData
```

```

 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2023, 1, 10, 12, 0, 0), EndTime = new
 DateTime(2023, 1, 10, 14, 0, 0) });
 return appData;
 }
 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

### Timeline views

When the feature **RowAutoHeight** is enabled in Timeline views, the row height gets auto-adjusted based on the number of overlapping events occupied on the same time range, which is demonstrated in the following example.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .RowAutoHeight(true)
 .SelectedDate(new DateTime(2019, 1, 10))
 .CurrentView(View.TimelineWeek)
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineDay
 },
 new ScheduleView { Option =
 Syncfusion.EJ2.Schedule.View.TimelineWeek },
 new ScheduleView { Option =
 Syncfusion.EJ2.Schedule.View.TimelineWorkWeek },
 new ScheduleView { Option =
 Syncfusion.EJ2.Schedule.View.TimelineMonth },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Agenda }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()

```

```

{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 6, Subject = "Mysteries of Bermuda Triangle", Location =
 "Bermuda", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 7, Subject = "Glaciers and Snowflakes", Location = "Himalayas",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 8, Subject = "Life on Mars", Location = "Space Centre USA",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 9, Subject = "Alien Civilization", Location = "Space Centre USA",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 10, Subject = "Wildlife Galleries", Location = "Africa",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 11, Subject = "Best Photography 2018", Location = "London",
 StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 12, Subject = "Smarter Puppies", Location = "Sweden", StartTime =
 new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new DateTime(2023, 1, 10, 11,
 0, 0) });
 appData.Add(new AppointmentData
 { Id = 13, Subject = "Myths of Andromeda Galaxy", Location = "Space
 Centre USA", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData

```

```

 { Id = 14, Subject = "Aliens vs Humans", Location = "Research Centre of
USA", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 15, Subject = "Facts of Humming Birds", Location = "California",
StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 16, Subject = "Sky Gazers", Location = "Alaska", StartTime = new
DateTime(2023, 1, 10, 9, 30, 0), EndTime = new DateTime(2023, 1, 10, 11, 0,
0) });
 appData.Add(new AppointmentData
 { Id = 17, Subject = "The Cycle of Seasons", Location = "Research Centre
of USA", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 18, Subject = "Space Galaxies and Planets", Location = "Space
Centre USA", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 19, Subject = "Lifecycle of Bumblebee", Location = "San
Francisco", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
DateTime(2023, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 20, Subject = "Alien Civilization", Location = "Space Centre
USA", StartTime = new DateTime(2023, 1, 10, 9, 30, 0), EndTime = new
DateTime(2023, 1, 10, 11, 0, 0) });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Timeline views with multiple resources

The following example shows how the auto row adjustment feature works on timeline views with multiple resources.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .RowAutoHeight(true)
 .SelectedDate(new DateTime(2018, 2, 11))
 .CurrentView(View.TimelineWeek)
 .Views(view => {
 view.Option(View.TimelineDay).Add();
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineWorkWeek).Add();
 view.Option(View.TimelineMonth).Add();
 }
)

```

```

 view.Option(View.Agenda).Add();
 })
 .Group(group =>
group.EnableCompactView(false).Resources(ViewBag.ResourceNames))
 .Resources(res => {
 res.DataSource(ViewBag.RoomDatas).Field("RoomId").Title("Room
Type").Name("MeetingRoom").TextField("name").IdField("id").ColorField("color
").AllowMultiple(true).Add();
 })
 .EventSettings(e => e.Fields(f =>
 f.Subject(sub => sub.Name("Subject").Title("Summary"))
 .Location(loc => loc.Name("Location").Title("Location"))
 .Description(des => des.Name("Description").Title("Comments"))
 .StartTime(st => st.Name("StartTime").Title("From"))
 .EndTime(et => et.Name("EndTime").Title("To"))
)
 .DataSource(ViewBag.datasources))
 .Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasources = GetScheduleData();
 List<RoomData> rooms = new List<RoomData>();
 rooms.Add(new RoomData { name = "Room A", id = 1, color = "#98AFC7" });
 rooms.Add(new RoomData { name = "Room B", id = 2, color = "#99c68e" });
 rooms.Add(new RoomData { name = "Room C", id = 3, color = "#C2B280" });
 rooms.Add(new RoomData { name = "Room D", id = 4, color = "#3090C7" });
 rooms.Add(new RoomData { name = "Room E", id = 5, color = "#95b9" });
 rooms.Add(new RoomData { name = "Room F", id = 6, color = "#95b9c7" });
 rooms.Add(new RoomData { name = "Room G", id = 7, color = "#deb887" });
 rooms.Add(new RoomData { name = "Room H", id = 8, color = "#3090C7" });
 rooms.Add(new RoomData { name = "Room I", id = 9, color = "#98AFC7" });
 rooms.Add(new RoomData { name = "Room J", id = 10, color = "#778899" });
 ViewBag.RoomDatas = rooms;
 string[] resources = new string[] { "MeetingRoom" };
 ViewBag.ResourceNames = resources;
 return View();
}

public class RoomData
{
 public int id { set; get; }
 public string name { set; get; }
 public string color { set; get; }
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 11, 9, 30, 0), EndTime = new DateTime(2023, 2, 11, 11, 0,
0), RoomId = 1 });
 appData.Add(new AppointmentData

```

```

 { Id = 2, Subject = "Thule Air Crash Report", StartTime = new
DateTime(2023, 2, 11, 9, 30, 0), EndTime = new DateTime(2023, 2, 11, 11, 0,
0), RoomId = 1 });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 11, 9, 30, 0), EndTime = new DateTime(2023, 2, 11, 11, 0, 0), RoomId = 1
});
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2018", StartTime = new
DateTime(2023, 2, 11, 9, 30, 0), EndTime = new DateTime(2023, 2, 11, 11, 0,
0), RoomId = 1 });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 11, 9, 0, 0), EndTime = new DateTime(2023, 2, 11, 11, 30,
0), RoomId = 2 });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public int RoomId { get; set; }
}

```

### Appointments occupying entire cell

By default, with the feature **RowAutoHeight**, there will be a space in the bottom of the cell when appointment is rendered. To avoid this space, we can set true to the property **IgnoreWhitespace** with in **EventSettings** whereas its default property value is false. In the following code example, the whitespace below the appointments has been ignored.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .RowAutoHeight(true)
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

 res.AllowMultiple(false).DataSource(ViewBag.Rooms).Field("RoomId").Title("Room").Name("Rooms").TextField("RoomText").IdField("Id").ColorField("RoomColor").Add();

 res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").GroupIDField("OwnerGroupId").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineMonth).Add();
 })
)

```

```

.EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource, IgnoreWhitespace = true})
.SelectedDate(new DateTime(2021, 7, 4))
.Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 // datasource for room resources
 List<RoomResource> rooms = new List<RoomResource>();
 rooms.Add(new RoomResource { RoomText = "ROOM 1", Id = 1, RoomColor =
"#cb6bb2" });
 rooms.Add(new RoomResource { RoomText = "ROOM 2", Id = 2, RoomColor =
"#56ca85" });
 ViewBag.Rooms = rooms;
 // datasource for owner resources
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerGroupId =
1, OwnerColor = "#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2,
OwnerGroupId = 2, OwnerColor = "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3,
OwnerGroupId = 1, OwnerColor = "#7499e1" });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Rooms", "Owners" };
 return View();
}

public List<ResourceData> GetResourceData()
{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2023, 7, 4, 9, 30, 0),
 EndTime = new DateTime(2023, 7, 4, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 1,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 7, 4, 10, 30, 0),
 EndTime = new DateTime(2023, 7, 4, 14, 0, 0),
 IsAllDay = false,
 OwnerId = 3,
 RoomId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,

```

```

 Subject = "Resource planning",
 StartTime = new DateTime(2023, 7, 5, 10, 0, 0),
 EndTime = new DateTime(2023, 7, 5, 12, 30, 0),
 IsAllDay = false,
 OwnerId = 2,
 RoomId = 2
 });
 return resourceData;
}
public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
 public int RoomId { get; set; }
}
public class RoomResource
{
 public string RoomText { set; get; }
 public int Id { set; get; }
 public string RoomColor { set; get; }
}
public class OwnerResource
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
 public int OwnerGroupId { set; get; }
}

```

**Note:** The property `IgnoreWhitespace` will be applicable only when `RowAutoHeight` feature is enabled in the Scheduler

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Header customization

The header part of Scheduler can be customized easily with the built-in options available.

#### Show or Hide header bar

By default, the header bar holds the date and view navigation options, through which the user can switch between the dates and various views. This header bar can be hidden from the UI by setting `false` to the `ShowHeaderBar` property. Its default value is `true`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")

```



```

 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .ShowHeaderBar(false)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Customizing header bar using template

Apart from the default date navigation and view options available on the header bar, you can add custom items into the Scheduler header bar by making use of the [ToolBarItems](#) property. To display the default items, it's essential to assign a [ToolBarName](#) field to each item. The names of the default items are **Previous**, **Next**, **Today**, **DateRangeText**, **NewEvent**, and **Views**. For custom items you can give the name as **Custom** to the name field. Here, the default items such as previous, next, date range text, and today have been used along with external icon as custom items.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@using Syncfusion.EJ2.Navigations;
@using Syncfusion.EJ2.DropDowns
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .Resources(res =>
 {
 res.DataSource(ViewBag.Calendars); res.Query("new
ej.data.Query().where('OwnerId', 'equal',

```

```

1)").Field("OwnerId").Title("Owners").Name("Owners").TextField("OwnerText").
IdField("OwnerId").ColorField("Color").AllowMultiple(true).Add();
 })
 .ToolBarItems(ViewBag.toolbarItems)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource,
 .Query("new ej.data.Query().where('OwnerId', 'equal', 1)") })
 .SelectedDate(new DateTime(2023, 10, 15))
 .Render()
)
<div style="width: 125px;">

@Html.EJS().DropDownList("dropelement").Width("125px").showClearButton(false
).Value("0").DataSource(ViewBag.Calendars).Fields(new
DropDownListFieldSettings { Text = "OwnerText", Value = "OwnerId"
}).Change("onResChange").Render()
</div>
<script type="text/javascript">
 function onResChange(args) {
 var value = args.value;
 var resourcePredicate;
 resourcePredicate = new ej.data.Predicate('OwnerId', 'equal',
value);
 scheduleObj.resources[0].query = resourcePredicate ? new
ej.data.Query().where(resourcePredicate) :
 new ej.data.Query().where('OwnerId', 'equal', 1);
 scheduleObj.eventSettings.query = new
ej.data.Query().where('OwnerId', 'equal', value);
 }
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 List<CalendarRes> ownerCollections = new List<CalendarRes>()
 {
 new CalendarRes { OwnerText = 'Margaret', OwnerId = 1, Color =
'#ea7a57' },
 new CalendarRes { OwnerText = 'Robert', OwnerId = 2, Color =
'#df5286' },
 new CalendarRes { OwnerText = 'Laura', OwnerId = 3, Color =
'#865fcf' }
 };
 ViewBag.Calendars = ownerCollections;
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 List<ScheduleToolBarItem> toolbarItems = new
List<ScheduleToolBarItem>()
 {
 new ScheduleToolBarItem { Name = ToolbarName.Previous, Align =
ItemAlign.Left },
 };
}

```

```

 new ScheduleToolBarItem { Name = ToolbarName.Next, Align =
ItemAlign.Left },
 new ScheduleToolBarItem { Name = ToolbarName.DateRangeText,
Align = ItemAlign.Left },
 new ScheduleToolBarItem { Template = "#dropelement", Type =
ItemType.Input, Align = ItemAlign.Center },
 new ScheduleToolBarItem { Name = ToolbarName.Today, Align =
ItemAlign.Right },
 };
 ViewBag.toolbarItems = toolbarItems;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData{
 Id: 1,
 Subject: 'Paris',
 StartTime: new Date(2023, 9, 29, 10, 0),
 EndTime: new Date(2023, 9, 29, 11, 30),
 IsAllDay: false,
 OwnerId: 1
 }),
 appData.Add(new AppointmentData{
 Id: 2,
 Subject: 'Meeting - 1',
 StartTime: new Date(2023, 9, 30, 10, 0),
 EndTime: new Date(2023, 9, 30, 12, 30),
 IsAllDay: false,
 OwnerId: 2
 }),
 appData.Add(new AppointmentData{
 Id: 3,
 Subject: 'Meeting - 2',
 StartTime: new Date(2023, 9, 30, 11, 0),
 EndTime: new Date(2023, 9, 30, 14, 30),
 IsAllDay: false,
 OwnerId: 3
 }),
 appData.Add(new AppointmentData{
 Id: 4,
 StartTime: new Date(2023, 9, 31),
 EndTime: new Date(2023, 10, 1),
 IsAllDay: true,
 OwnerId: 1
 }),
 appData.Add(new AppointmentData{
 Id: 5,
 Subject: 'Conference - 2',
 StartTime: new Date(2023, 9, 31, 22, 0),
 EndTime: new Date(2023, 10, 1, 0, 0),
 IsAllDay: false,
 OwnerId: 2
 }),
 appData.Add(new AppointmentData{
 Id: 6,
 Subject: 'Conference - 3',

```

```
 StartTime: new Date(2023, 10, 1, 9, 30),
 EndTime: new Date(2023, 10, 1, 11, 45),
 IsAllDay: false,
 OwnerId: 3
)),
 appData.Add(new AppointmentData{
 Id: 7,
 Subject: 'Conference - 4',
 StartTime: new Date(2023, 10, 1, 10, 30),
 EndTime: new Date(2023, 10, 1, 12, 45),
 IsAllDay: false,
 OwnerId: 1
)),
 appData.Add(new AppointmentData{
 Id: 8,
 Subject: 'Travelling',
 StartTime: new Date(2023, 10, 1, 11, 30),
 EndTime: new Date(2023, 10, 1, 13, 45),
 IsAllDay: false,
 OwnerId: 2
)),
 appData.Add(new AppointmentData{
 Id: 9,
 Subject: 'Vacation',
 StartTime: new Date(2023, 10, 2, 10, 0),
 EndTime: new Date(2023, 10, 2, 12, 30),
 IsAllDay: false,
 OwnerId: 3
)),
 appData.Add(new AppointmentData{
 Id: 10,
 Subject: 'Conference',
 StartTime: new Date(2023, 10, 2, 15, 30),
 EndTime: new Date(2023, 10, 2, 18, 45),
 IsAllDay: false,
 OwnerId: 1
)),
 appData.Add(new AppointmentData{
 Id: 11,
 Subject: 'Vacation',
 StartTime: new Date(2023, 10, 3, 10, 15),
 EndTime: new Date(2023, 10, 3, 14, 45),
 IsAllDay: false,
 OwnerId: 2
)),
 appData.Add(new AppointmentData{
 Id: 12,
 Subject: 'Conference',
 StartTime: new Date(2023, 10, 4, 9, 30),
 EndTime: new Date(2023, 10, 5, 5, 45),
 IsAllDay: false,
 OwnerId: 3
)),
 appData.Add(new AppointmentData{
 Id: 13,
 StartTime: new Date(2023, 10, 5, 10, 0),
 EndTime: new Date(2023, 10, 5, 11, 30),
```

```
 IsAllDay: false,
 OwnerId: 1
)),
 appData.Add(new AppointmentData{
 Id: 14,
 Subject: 'Same Time',
 StartTime: new Date(2023, 10, 5, 10, 0),
 EndTime: new Date(2023, 10, 5, 11, 30),
 IsAllDay: false,
 OwnerId: 2
)),
 appData.Add(new AppointmentData{
 Id: 15,
 Subject: 'Same Time',
 StartTime: new Date(2023, 10, 5, 10, 0),
 EndTime: new Date(2023, 10, 5, 11, 30),
 IsAllDay: false,
 OwnerId: 3
)),
 appData.Add(new AppointmentData{
 Id: 16,
 Subject: 'Same Time',
 StartTime: new Date(2023, 10, 5, 10, 0),
 EndTime: new Date(2023, 10, 5, 11, 30),
 IsAllDay: false,
 OwnerId: 1
)),
 appData.Add(new AppointmentData{
 Id: 17,
 Subject: 'Same Time',
 StartTime: new Date(2023, 10, 5, 10, 0),
 EndTime: new Date(2023, 10, 5, 11, 30),
 IsAllDay: false,
 OwnerId: 2
)),
 appData.Add(new AppointmentData{
 Id: 18,
 Subject: 'Same Time',
 StartTime: new Date(2023, 10, 5, 10, 0),
 EndTime: new Date(2023, 10, 5, 11, 30),
 IsAllDay: false,
 OwnerId: 3
)),
 appData.Add(new AppointmentData{
 Id: 19,
 Subject: 'Meeting - 1',
 StartTime: new Date(2023, 10, 6),
 EndTime: new Date(2023, 10, 7),
 IsAllDay: true,
 OwnerId: 1
)),
 appData.Add(new AppointmentData
 Id: 20,
 Subject: 'Meeting - 2',
 StartTime: new Date(2023, 10, 6, 11, 0),
 EndTime: new Date(2023, 10, 6, 14, 30),
 IsAllDay: false,
```

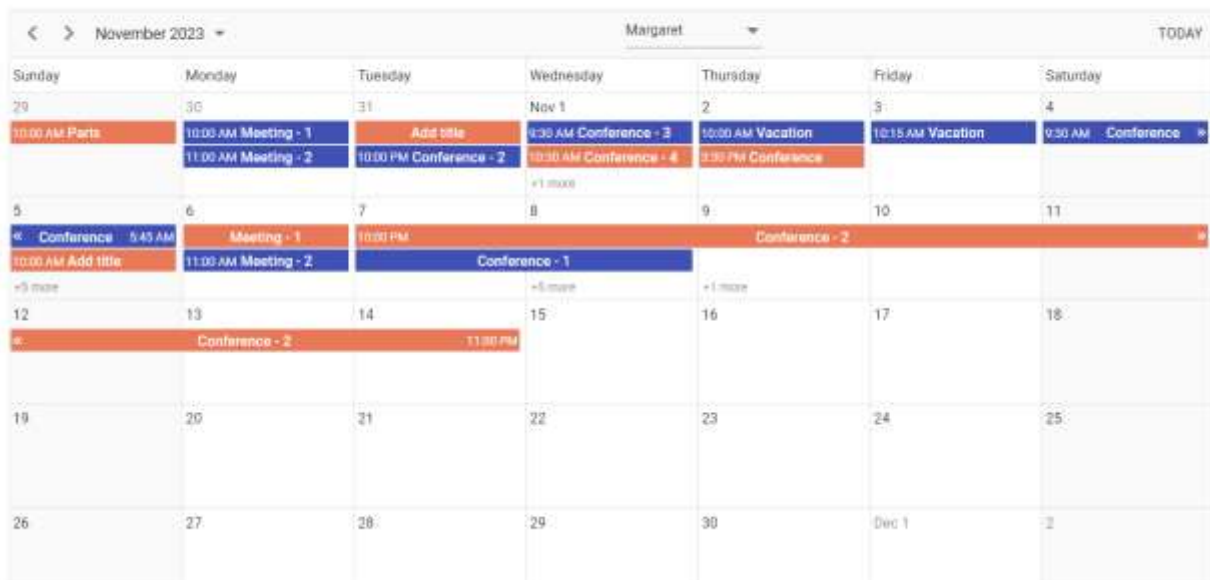
```
 OwnerId: 3
 }},
 appData.Add(new AppointmentData{
 Id: 21,
 Subject: 'Conference - 1',
 StartTime: new Date(2023, 10, 7, 22, 0),
 EndTime: new Date(2023, 10, 8, 20, 0),
 IsAllDay: true,
 OwnerId: 2
 }},
 appData.Add(new AppointmentData{
 Id: 22,
 Subject: 'Conference - 2',
 StartTime: new Date(2023, 10, 7, 22, 0),
 EndTime: new Date(2023, 10, 14, 23, 0),
 IsAllDay: false,
 OwnerId: 1
 }},
 appData.Add(new AppointmentData{
 Id: 23,
 Subject: 'Conference - 3',
 StartTime: new Date(2023, 10, 8, 9, 30),
 EndTime: new Date(2023, 10, 9, 11, 45),
 IsAllDay: true,
 OwnerId: 2
 }},
 appData.Add(new AppointmentData{
 Id: 24,
 Subject: 'Conference - 3 - A',
 StartTime: new Date(2023, 10, 8, 9, 30),
 EndTime: new Date(2023, 10, 8, 10, 0),
 IsAllDay: true,
 OwnerId: 3
 }},
 appData.Add(new AppointmentData{
 Id: 25,
 Subject: 'Conference - 3 - B',
 StartTime: new Date(2023, 10, 8, 10, 0),
 EndTime: new Date(2023, 10, 8, 10, 30),
 IsAllDay: false,
 OwnerId: 1
 }},
 appData.Add(new AppointmentData{
 Id: 26,
 Subject: 'Conference - 4',
 StartTime: new Date(2023, 10, 8, 10, 30),
 EndTime: new Date(2023, 10, 8, 12, 45),
 IsAllDay: false,
 OwnerId: 2
 }},
 appData.Add(new AppointmentData{
 Id: 27,
 Subject: 'Travelling',
 StartTime: new Date(2023, 10, 8, 11, 30),
 EndTime: new Date(2023, 10, 8, 13, 45),
 IsAllDay: false,
 OwnerId: 3
 })
```

```

 })
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```



### Customizing header bar using events

Apart from the default date navigation and view options available on the header bar, you can add custom items into the Scheduler header bar by making use of the `ActionBegin` event. Here, an employee image is added to the header bar, clicking on which will open the popup showing that person's short profile information.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .ActionComplete("onActionComplete")
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

```

<style>
 .e-schedule .e-schedule-toolbar .user-icon {
 background-image:
url('https://ej2.syncfusion.com/demos/src/schedule/images/nancy.png');
 background-position: center center;
 background-repeat: no-repeat;
 background-size: cover;
 border-radius: 50%;
 height: 24px;
 min-width: 24px !important;
 width: 24px !important;
 }
 .e-schedule .e-schedule-toolbar .e-toolbar-items span.e-btn-icon.user-
icon.e-icons {
 line-height: 24px !important;
 min-height: 24px !important;
 }
 .e-schedule .e-schedule-toolbar .e-toolbar-items .e-schedule-user-icon
.e-tbar-btn:hover {
 background-color: inherit;
 }
 .e-schedule .e-schedule-toolbar .e-toolbar-items .e-schedule-user-icon
.e-tbar-btn-text {
 display: none;
 }
 .e-schedule .e-schedule-toolbar .e-toolbar-pop .e-schedule-user-icon .e-
tbar-btn-text {
 padding-left: 8px !important;
 }
 .e-profile-wrapper {
 width: 210px;
 height: 80px;
 background-color: #fafafa;
 box-shadow: inset 0 0 5px rgba(0, 0, 0, 0.2);
 overflow: hidden;
 }
 .e-profile-wrapper .profile-container {
 display: flex;
 padding: 10px;
 }
 .e-profile-wrapper .profile-image {
 background-image:
url('https://ej2.syncfusion.com/demos/src/schedule/images/nancy.png');
 background-position: center center;
 background-repeat: no-repeat;
 background-size: cover;
 border-radius: 50%;
 box-shadow: inset 0 0 1px #e0e0e0, inset 0 0 14px rgba(0, 0, 0,
0.2);
 width: 60px;
 height: 60px;
 }
 .e-profile-wrapper .content-wrap {
 padding-left: 10px;
 }
 .e-profile-wrapper .name {
 font-size: 14px;
 }

```



```

 line-height: 20px;
 font-weight: 500;
 margin-top: 2px;
 }
 .e-profile-wrapper .destination {
 font-size: 12px;
 }
 .e-profile-wrapper .status-icon {
 height: 6px;
 width: 6px;
 background: green;
 border-radius: 100%;
 float: left;
 margin-right: 4px;
 margin-top: 4px;
 }
 .e-profile-wrapper .status {
 font-size: 11px;
 }
</style>
<script type="text/javascript">
 var profilePopup;
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var userIconItem = {
 align: 'Right', prefixIcon: 'user-icon', text: 'Nancy',
 cssClass: 'e-schedule-user-icon'
 };
 args.items.push(userIconItem);
 }
 }
 function onActionComplete(args) {
 var scheduleElement = document.getElementById('schedule');
 if (args.requestType === 'toolBarItemRendered') {
 var userIconEle = scheduleElement.querySelector('.e-schedule-
user-icon');
 userIconEle.onclick = function () {
 profilePopup.relateTo = userIconEle;
 profilePopup.dataBind();
 if (profilePopup.element.classList.contains('e-popup-
close')) {
 profilePopup.show();
 } else {
 profilePopup.hide();
 }
 };
 }
 var userContentEle = ej.base.createElement('div', {
 className: 'e-profile-wrapper'
 });
 scheduleElement.parentElement.appendChild(userContentEle);
 var userIconEle = scheduleElement.querySelector('.e-schedule-user-
icon');
 var getDOMString = ej.base.compile('<div class="profile-
container"><div class="profile-image">' +
 '</div><div class="content-wrap"><div class="name">Nancy</div>'
+

```

```

 '<div class="destination">Product Manager</div><div
class="status">' +
 '<div class="status-icon"></div>Online</div></div></div>');
 var output = getDOMString({});
 profilePopup = new ej.popups.Popup(userContentEle, {
 content: output[0],
 relateTo: userIconEle,
 position: { X: 'left', Y: 'bottom' },
 collision: { X: 'flip', Y: 'flip' },
 targetType: 'relative',
 viewPortElement: scheduleElement,
 width: 210,
 height: 80
 });
 profilePopup.hide();
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### How to display the view options within the header bar popup

By default, the header bar holds the view navigation options, through which the user can switch between various views. You can move this view options to the header bar popup by setting `true` to the `EnableAdaptiveUI` property.

## CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .EnableAdaptiveUI(true)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

## DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

**Note:** Refer [here](#) to know more about adaptive UI in resources scheduler.

### Date header customization

The Scheduler UI that displays the date text on all views are considered as the date header cells. You can customize the date header cells of Scheduler either using `DateHeaderTemplate` or `RenderCell` event.

#### Using date header template

The `DateHeaderTemplate` option is used to customize the date header cells of day, week and work-week views.

## CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
```

```

 .DateHeaderTemplate("#template")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
 <style>
 .weather-text {
 font-size: 11px;
 }
 </style>
 <script id="template" type="text/template">
 <div class="date-text">${getDateHeaderText(data.date)}</div>
 ${getWeather(data.date)}
 </script>
 <script type="text/javascript">
 var instance = new ej.base.Internationalization();
 window.getDateHeaderText = function (value) {
 return instance.formatDate(value, { skeleton: 'Ed' });
 };
 function getWeather(value) {
 switch (value.getDay()) {
 case 0:
 return '<div class="weather-text">25°C</div>';
 case 1:
 return '<div class="weather-text">18°C</div>';
 case 2:
 return '<div class="weather-text">10°C</div>';
 case 3:
 return '<div class="weather-text">16°C</div>';
 case 4:
 return '<div class="weather-text">8°C</div>';
 case 5:
 return '<div class="weather-text">27°C</div>';
 case 6:
 return '<div class="weather-text">17°C</div>';
 default:
 return null;
 }
 }
 </script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{

```

```

List<AppointmentData> appData = new List<AppointmentData>();
appData.Add(new AppointmentData
{
 Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Using renderCell event

In month view, the date header template is not applicable and therefore the same customization can be added beside the date text in month cells by making use of the `RenderCell` event.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .RenderCell("onRenderCell")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .weather-text {
 font-size: 11px;
 }
</style>
<script type="text/javascript">
 function getWeather(value) {
 switch (value.getDay()) {
 case 0:
 return '<div class="weather-text">25°C</div>';
 case 1:
 return '<div class="weather-text">18°C</div>';
 case 2:
 return '<div class="weather-text">10°C</div>';
 case 3:
 return '<div class="weather-text">16°C</div>';
 case 4:
 return '<div class="weather-text">8°C</div>';
 case 5:
 return '<div class="weather-text">27°C</div>';
 case 6:
 return '<div class="weather-text">17°C</div>';
 default:

```

```

 return null;
 }
}
function onRenderCell(args) {
 if (args.elementType === 'monthCells') {
 var ele = document.createElement('div');
 ele.innerHTML = getWeather(args.date);
 (args.element).appendChild(ele.firstChild);
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Customizing the date range text

The **DateRangeTemplate** option allows you to customize the text content of the date range displayed in the scheduler. By default, the date range text is determined by the scheduler view being used.

However, you can use the **DateRangeTemplate** option to override the default text and specify your own custom text to be displayed.

The **DateRangeTemplate** property includes **startDate**, **endDate** and **currentView** options, you can customize the date range text using these available options.

## CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")

```

```

 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .DateRangeTemplate("#template")
 .Render()
)
<script id="template" type="text/template">
 <div class="date-text">${getDateRange(data.date)}</div>
</script>
<script type="text/javascript">
 var instance = new ej.base.Internationalization();
 window.getDateRange = function (value) {
 return value.toLocaleString('en-us', { month: 'long' }) + ' ' +
value.getFullYear();
 };
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## TimeScale Customization

The time slots are usually the time cells that are displayed on the Day, Week and Work Week views of both the calendar (to the left most position) and timeline views (at the top position). The **TimeScale** property allows you to control and set the required time slot duration for the work cells displayed on Scheduler. It includes the following sub-options such as,

- **Enable** - When set to **true**, allows the Scheduler to display the appointments accurately against the exact time duration. If set to **false**, all the appointments of a day will be displayed one below the other with no grid lines displayed. Its default value is **true**.
- **Interval** – Defines the time duration on which the time axis to be displayed either in 1 hour or 30 minutes interval and so on. It accepts the values in minutes and defaults to 60.
- **SlotCount** – Decides the number of slot count to be split for the specified time interval duration. It defaults to 2, thus displaying two slots to represent an hour(each slot depicting 30 minutes duration).

Note: The upper limit for rendering slots within a single day, utilizing the **Interval** and **SlotCount** properties of the **TimeScale**, stands at 1000. This constraint aligns with the maximum **colspan** value permissible for the **table** element, also capped at 1000. This particular restriction is relevant exclusively to the **TimelineDay**, **TimelineWeek** and **TimelineWorkWeek** views.

### Setting different time slot duration

The **Interval** and **SlotCount** properties can be used together on the Scheduler to set different time slot duration which is depicted in the following code example. Here, six time slots together represents an hour.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .TimeScale(ts => ts.Enable(true).Interval(60).SlotCount(6))
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Render()
)
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{

```



```

List<AppointmentData> appData = new List<AppointmentData>();
appData.Add(new AppointmentData
{ Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 9, 30, 0), EndTime = new DateTime(2023, 2, 13, 11, 0, 0) });
appData.Add(new AppointmentData
{ Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Customizing time cells using template

The **TimeScale** property also provides template option to allow customization of time slots which are as follows,

- **MajorSlotTemplate** - The template option to be applied for major time slots. Here, the template accepts either the string or **HTMLElement** as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.
- **MinorSlotTemplate** - The template option to be applied for minor time slots. Here, the template accepts either the string or **HTMLElement** as template design and then the parsed design is displayed onto the time cells. The time details can be accessed within this template.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .TimeScale(ts =>
 ts.Enable(true)
 .Interval(60)
 .SlotCount(6)
 .MajorSlotTemplate("#majorSlotTemplate")
 .MinorSlotTemplate("#minorSlotTemplate")
)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Render()
)
<script id="majorSlotTemplate" type="text/x-template">

```

```

<div>${majorSlotTemplate(data.date)}</div>
</script>
<script id="minorSlotTemplate" type="text/x-template">
 <div style="text-align: right; margin-right:
15px">${minorSlotTemplate(data.date)}</div>
</script>
<script type="text/javascript">
 window.majorSlotTemplate = function (date) {
 var instance = new ej.base.Internationalization();
 return instance.formatDate(date, { skeleton: 'hm' });
 };
 window.minorSlotTemplate = function (date) {
 var instance = new ej.base.Internationalization();
 return instance.formatDate(date, { skeleton: 'ms' }).replace(':00',
'');
 };
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 9, 30, 0), EndTime = new DateTime(2023, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Hide the timescale

The grid lines which indicates the exact time duration can be enabled or disabled on the Scheduler, by setting `true` or `false` to the `Enable` option within the `TimeScale` property. It's default value is `true`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
.Width("100%")

```

```

 .Height("550px")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .TimeScale(ts => ts.Enable(false))
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 9, 30, 0), EndTime = new DateTime(2023, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Highlighting current date and time

By default, Scheduler indicates current date with a highlighted date header on all views, as well as marks accurately the system's current time on specific views such as Day, Week, Work Week, Timeline Day, Timeline Week and Timeline Work Week views. To stop highlighting the current time indicator on Scheduler views, set `false` to the `ShowTimeIndicator` property which defaults to `true`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 })
)

```

```

 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 view.Option(View.TimelineDay).Add();
 view.Option(View.TimelineWeek).Add();
 view.Option(View.TimelineWorkWeek).Add();
 })
 .ShowTimeIndicator(false)
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Setting working days and hours

The Scheduler can be customized on various aspects as well as it inherits almost all the calendar-specific features such as options,

- To set custom time range display on Scheduler
- To set different working hours
- To set different working days
- To set different first day of week
- To show/hide weekend days
- To show the week number

### Set working days

By default, Scheduler considers the week days from Monday to Friday as **Working days** and therefore defaults to [1,2,3,4,5] - where 1 represents Monday, 2 represents Tuesday and so on. The days which are not defined in this working days collection are considered as non-working days. Therefore, when the weekend days are set to hide from Scheduler, all those non-working days too get hidden from the layout.

The Work week and Timeline Work week views display exactly the defined working days on Scheduler layout, whereas other views display all the days and simply differentiate the non-working days on UI with inactive cell color.

**Note:** The working or business hours depiction on Scheduler are usually valid only on these specified working days.

The following example code depicts how to set the Scheduler to display Monday, Wednesday and Friday as working days of a week.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
```

```
@(Html.EJS().Schedule("schedule")
.Width("100%")
.Height("550px")
.Views(view => {
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 view.Option(View.Month).Add();
}))
.EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
.WorkDays(ViewBag.workday)
.SelectedDate(new DateTime(2018, 2, 15))
.Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workday = new int[] { 1, 3, 5 };
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 9, 30, 0), EndTime = new DateTime(2023, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Hiding weekend days

The **ShowWeekend** property is used to either show or hide the weekend days of a week and it is not applicable on Work week view (as non-working days are usually not displayed on work week view). By default, it is set to **true**. The days which are not a part of the working days collection of a Scheduler are usually considered as non-working or weekend days.

Here, the working days are defined as [1, 3, 4, 5] on Scheduler and therefore the remaining days (0, 2, 6 – Sunday, Tuesday and Saturday) are considered as non-working or weekend days and will be hidden from all the views when **ShowWeekend** property is set to **false**.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.TimelineMonth).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .ShowWeekend(false)
 .WorkDays(ViewBag.workday)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workday = new int[] { 1, 3, 4, 5 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 12, 9, 30, 0), EndTime = new DateTime(2023, 2, 12, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

**Show week numbers**

It is possible to show the week number count of a week in the header bar of the Scheduler by setting true to **ShowWeekNumber** property. By default, its default value is **false**. In Month view, the week numbers are displayed as a first column.

**Note:** The **ShowWeekNumber** property is not applicable on Timeline views, as it has the equivalent [HeaderRows](#) property to handle such requirement with additional customization.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .ShowWeekNumber(true)
 .WorkDays(ViewBag.workday)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workday = new int[] { 1, 3, 4, 5 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 12, 9, 30, 0), EndTime = new DateTime(2023, 2, 12, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

*Different options in showing week numbers*

By default, week numbers are shown in the Scheduler based on the first day of the year. However, the week numbers can be determined based on the following criteria.

**FirstDay** – The first week of the year is calculated based on the first day of the year.

**FirstFourDayWeek** – The first week of the year begins from the first week with four or more days.

**FirstFullWeek** – The first week of the year begins when meeting the first day of the week (firstDayOfWeek) and the first day of the year.

For more details refer to [this link](#)

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .ShowWeekNumber(true)
 .WeekRule(WeekRule.FirstFourDayWeek)
 .WorkDays(ViewBag.workday)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 ViewBag.workday = new int[] { 1, 3, 5 };
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 9, 30, 0), EndTime = new DateTime(2018, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```



**Note:** Enable the `showWeekNumber` property to configure the `weekRule` property. Also, the `weekRule` property depends on the value of the `firstDayOfWeek` property.

### Set working hours

Working hours indicates the work hour limit within the Scheduler, which is visually highlighted with an active color on work cells. The working hours can be set on Scheduler using the `WorkHours` property which is of object type and includes the following sub-options,

- **Highlight** – enables/disables the highlighting of work hours.
- **Start** - sets the start time of the working/business hour of a day.
- **End** - sets the end time limit of the working/business hour of a day.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .WorkHours(wh =>
 wh.Highlight(true)
 .Start("11:00")
 .End("20:00")
)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 13, 11, 0, 0), EndTime = new DateTime(2023, 2, 13, 13, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 15, 11, 0, 0), EndTime = new DateTime(2023, 2, 15, 13, 0,
0) });
 return appData;
}
```

```
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Scheduler displaying custom hours

It is possible to display the event Scheduler layout with specific time durations by hiding the unwanted hours. To do so, set the start and end hour for the Scheduler using the `StartHour` and `EndHour` properties respectively.

The following code example displays the Scheduler starting from the time range 7.00 AM to 6.00 PM and the remaining hours are hidden on the UI.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .StartHour("07:00")
 .EndHour("18:00")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 12, 9, 30, 0), EndTime = new DateTime(2023, 2, 12, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}
```

```
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Setting start day of the week

By default, Scheduler defaults to **Sunday** as its first day of a week. To change the Scheduler's start day of a week with different day, set the **FirstDayOfWeek** property with the values ranging from 0 to 6.

Here, Sunday is always denoted as 0, Monday as 1 and so on.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .FirstDayOfWeek(1)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 12, 9, 30, 0), EndTime = new DateTime(2023, 2, 12, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
}
```

```

public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}

```

### Scroll to specific time and date

You can manually scroll to a specific time on Scheduler by making use of the `scrollTo` method as depicted in the following code example.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
<table style="width: 100%">
 <tbody>
 <tr>
 <td>
 <div>
 Scroll To
 </div>
 </td>
 <td>
 <div>
 @(Html.EJS().TimePicker("ScrollToHour")
 .Width("120")
 .Change("onChange")
 .Value(new DateTime(2000, 1, 1, 9, 0, 0))
 .Format("HH:mm")
 .Render())
 </div>
 </td>
 </tr>
 </tbody>
</table>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .Render())
</div>
<script type="text/javascript">
 function onChange(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.scrollTo(args.text);
 }
</script>

```

#### DATA.CS

```

public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()

```

```

{
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
};
ViewBag.view = viewOption;
return View();
}

```

#### *How to scroll to current time on initial load*

There are scenarios where you may need to load the Scheduler displaying the system's current time on the currently visible view port area. In such cases, the Scheduler needs to be scrolled to a specific time based on the system's current time which is depicted in the following code example.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(view => {
 view.Option(View.Day).Add();
 view.Option(View.Week).Add();
 view.Option(View.WorkWeek).Add();
 })
 .Created("onCreate")
 .Render()
)
<script type="text/javascript">
 function onCreate() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var curTime = new Date();
 var hours = curTime.getHours() < 10 ? '0'
+curTime.getHours().toString() : curTime.getHours().toString();
 var minutes = curTime.getMinutes().toString();
 var time = hours + ':' + minutes;
 scheduleObj.scrollTo(time);
 }
</script>

```

#### **DATA.CS**

```

public ActionResult Index()
{
 return View();
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## See Also

- [To display the current time indicator](#)
- [To set different working hours dynamically](#)
- [To set different working hours for each resources](#)
- [To set different working days for each resources](#)

## Cell Customization

The cells of the Scheduler can be easily customized either using the cell template or **RenderCell** event.

## Setting cell dimensions in all views

The height and width of the Scheduler cells can be customized either to increase or reduce its size through the **CssClass** property, which overrides the default CSS applied on cells.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .CssClass("schedule-cell-dimension")
 .Height("550px")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<style>
 .schedule-cell-dimension.e-schedule .e-vertical-view .e-date-header-wrap
table col,
 .schedule-cell-dimension.e-schedule .e-vertical-view .e-content-wrap
table col {
 width: 200px;
 }
 .schedule-cell-dimension.e-schedule .e-vertical-view .e-time-cells-wrap
table td,
 .schedule-cell-dimension.e-schedule .e-vertical-view .e-work-cells {
 height: 100px;
 }
 .schedule-cell-dimension.e-schedule .e-month-view .e-work-cells,
 .schedule-cell-dimension.e-schedule .e-month-view .e-date-header-wrap
table col {
 width: 200px;
 }
 .schedule-cell-dimension.e-schedule .e-month-view .e-work-cells {
 height: 200px;
 }
</style>
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetEmployeeEventData();
}
```

```

List<ScheduleView> viewOption = new List<ScheduleView>()
{
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.Month }
};
ViewBag.view = viewOption;
return View();
}
public List<EmployeeEventData> GetEmployeeEventData()
{
 List<EmployeeEventData> employeeEventData = new
List<EmployeeEventData>();
 employeeEventData.Add(new EmployeeEventData
 { Id = 1,
 Subject = "Project Workflow Analysis",
 StartTime = new DateTime(2023, 2, 14, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 14, 11, 0, 0),
 CategoryColor = "#1aaa55"
 });
 employeeEventData.Add(new EmployeeEventData
 {
 Id = 2,
 Subject = "Project Requirement Planning",
 StartTime = new DateTime(2023, 2, 15, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 11, 0, 0),
 CategoryColor = "#357cd2"
 });
 employeeEventData.Add(new EmployeeEventData
 {
 Id = 3,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2023, 2, 16, 9, 0, 0),
 EndTime = new DateTime(2023, 2, 16, 11, 0, 0),
 CategoryColor = "#7fa900"
 });
 return employeeEventData;
}
public class EmployeeEventData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public string CategoryColor { get; set; }
}

```

### Check for cell availability

You can check whether the given time range slots are available for event creation or already occupied by other events using the `isSlotAvailable` method. In the following code example, if a specific time slot already contains an appointment, then no more appointments can be added to that cell.

Note: The **isSlotAvailable** is centered around verifying appointments within the present view's date range. Yet, it does not encompass an evaluation of availability for recurrence occurrences that fall beyond this particular date range.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onActionBegin(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 if (args.requestType === 'eventCreate' && args.data.length > 0) {
 var eventData = args.data[0];
 var eventField = scheduleObj.eventFields;
 var startDate = eventData[eventField.startTime];
 var endDate = eventData[eventField.endTime];
 args.cancel = !scheduleObj.isSlotAvailable(startDate, endDate);
 }
 }
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView {Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2023,
2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Meteor Showers in 2018", StartTime = new
DateTime(2023, 2, 15, 9, 30, 0), EndTime = new DateTime(2023, 2, 15, 11, 0,
0) });
 appData.Add(new AppointmentData
```



```

 { Id = 3, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2023, 2, 16, 9, 30, 0), EndTime = new DateTime(2023, 2, 16, 11, 0,
0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

### Customizing cells in all the views

It is possible to customize the appearance of the cells using both template options and **RenderCell** event on all the views.

#### Using template

The **CellTemplate** option accepts the template string and is used to customize the cell background with specific images or appropriate text on the given date values.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@{
 var template = "${if(type === 'monthCells')}<div
class='templatewrap'>${getCellContent(data.date)}</div>${/if}";
}
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Views(ViewBag.view)
 .Height("550px")
 .CellTemplate(@template)
 .SelectedDate(new DateTime(2017, 12, 15))
 .Render()
)
<style>
 .e-schedule .e-month-view .e-work-cells {
 position: relative;
 }
 .e-schedule .templatewrap {
 text-align: center;
 position: absolute;
 width: 100%;
 }
 .e-schedule .templatewrap img {
 width: 25px;
 height: 25px;
 }
 .e-schedule .caption {
 overflow: hidden;
 text-overflow: ellipsis;
 vertical-align: middle;
 }
</style>

```

```

<script type="text/javascript">
 function getCellContent(date) {
 if (date.getMonth() === 9 && date.getDate() === 31) {
 return '<div class="caption">Thanksgiving day</div>';
 } else if (date.getMonth() === 11 && date.getDate() === 9) {
 return '<div class="caption">Party time</div>';
 } else if (date.getMonth() === 11 && date.getDate() === 13) {
 return '<div class="caption">Party time</div>';
 } else if (date.getMonth() === 11 && date.getDate() === 22) {
 return '<div class="caption">Happy birthday</div>';
 } else if (date.getMonth() === 11 && date.getDate() === 24) {
 return '<div class="caption">Christmas Eve</div>';
 } else if (date.getMonth() === 11 && date.getDate() === 25) {
 return '<div class="caption">Christmas Day</div>';
 } else if (date.getMonth() === 0 && date.getDate() === 1) {
 return '<div class="caption">New Year"s Day</div>';
 } else if (date.getMonth() === 0 && date.getDate() === 14) {
 return '<div class="caption">Get together</div>';
 }
 return '';
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

```

### Using renderCell event

An alternative to **CellTemplate** is the **RenderCell** event, which can also be used to customize the cells with appropriate images or formatted text values.

## CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Views(ViewBag.view)
 .Height("550px")
 .RenderCell("onRenderCell")
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onRenderCell(args) {
 if (args.element.classList.contains('e-work-hours') &&
 (args.date.getDay() === 1)) {
 args.element.style.background = '#1aaa55';
 }
 }

```

```

 } else if (args.element.classList.contains('e-work-hours') &&
(args.date.getDay() === 2)) {
 args.element.style.background = '#357cd2';
 } else if (args.element.classList.contains('e-work-hours') &&
(args.date.getDay() === 3)) {
 args.element.style.background = '#e2ff89';
 } else if (args.element.classList.contains('e-work-hours') &&
(args.date.getDay() === 4)) {
 args.element.style.background = '#f57f17';
 } else if (args.element.classList.contains('e-work-hours') &&
(args.date.getDay() === 5)) {
 args.element.style.background = '#00bdae';
 }
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}

```

You can customize cells such as work cells, month cells, all-day cells, header cells, resource header cells using `RenderCell` event by checking the `elementType` option within the event. You can check `elementType` with any of the following.

| Element type                    | Description                                              |
|---------------------------------|----------------------------------------------------------|
| ----- -----                     |                                                          |
| <code>dateHeader</code>         | triggers on header cell rendering.                       |
| <code>monthDay</code>           | triggers on header cell in month view rendering.         |
| <code>resourceHeader</code>     | triggers on resource header cell rendering.              |
| <code>alldayCells</code>        | triggers on all day cell rendering.                      |
| <code>emptyCells</code>         | triggers on empty cell rendering on header bar.          |
| <code>resourceGroupCells</code> | triggers on rendering of work cells for parent resource. |
| <code>workCells</code>          | triggers on work cell rendering.                         |
| <code>monthCells</code>         | triggers on month cell rendering.                        |
| <code>majorSlot</code>          | triggers on major time slot cell rendering.              |
| <code>minorSlot</code>          | triggers on minor time slot cell rendering.              |

| `weekNumberCell` | triggers on cell displaying week number. |

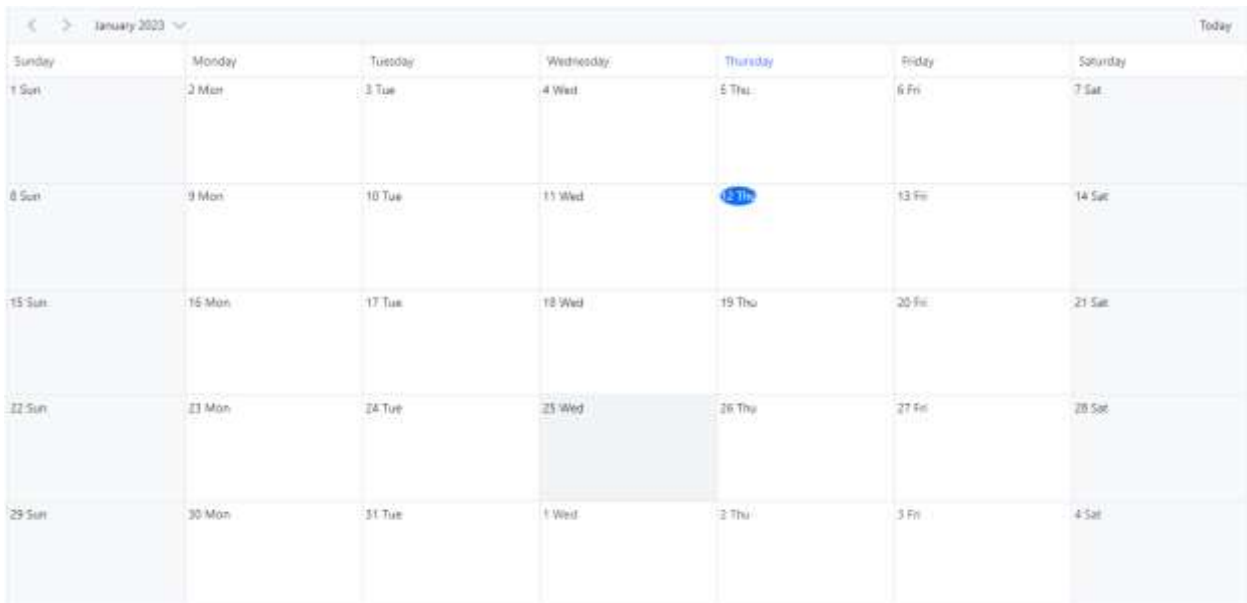
### Customizing cell header in month view

The month header of each date cell in the month view can be customized using the `CellHeaderTemplate` option which accepts the string or `HTMLElement`. The corresponding date can be accessed with the template.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@{
 var template = "<div>${getCellHeaderContent(data.date)}</div>";
}
@(Html.EJS().Schedule("schedule")
 .Views(view => { view.Option(View.Month).Add(); })
 .Height("550px")
 .CellHeaderTemplate(@template)
 .Render()
)
<script type="text/javascript">
 var instance = new ej.base.Internationalization();
 function getCellHeaderContent(date) {
 return instance.formatDate(date, { skeleton: "Ed" });
 }
</script>
```

#### CELL-HEADER-CUSTOMIZATION.CS



### Customizing the minimum and maximum date values

Providing the `minDate` and `maxDate` property with some date values, allows the Scheduler to set the minimum and maximum date range. The Scheduler date that lies beyond this minimum and maximum

date range will be in a disabled state so that the date navigation will be blocked beyond the specified date range.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Views(ViewBag.view)
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 17))
 .minDate(new DateTime(2017, 4, 17))
 .maxDate(new DateTime(2018, 5, 17))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.TimelineDay }
 };
 ViewBag.view = viewOption;
 return View();
}
```

**Note:** By default, the `minDate` property value is set to new Date(1900, 0, 1) and `maxDate` property value is set to new Date(2099, 11, 31). The user can also set the customized `minDate` and `maxDate` property values.

### How to disable multiple cell and row selection in Schedule

By default, the `AllowMultiCellSelection` and `AllowMultiRowSelection` properties of the Schedule are set to `true`. So, the Schedule allows user to select multiple cells and rows. If the user want to disable this multiple cell and row selection. The user can disable this feature by setting up `false` to these properties.

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Set state persistence

State persistence allowed Scheduler to retain the `CurrentView`, `SelectedDate` and Scroll position values in the `localStorage` for state maintenance even if the browser is refreshed or if you move to the next page within the browser. This action is handled through the `EnablePersistence` property which is set to `false` by default. When it is set to `true`, `CurrentView`, `SelectedDate` and Scroll position values of the scheduler component will be retained even after refreshing the page.

**Note:** Scheduler `id` is essential to set state persistence.

The following sample demonstrates how to set state persistence of the Scheduler component.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .EnablePersistence(true)
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 10, 9, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 10, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 10, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 10, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 10, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 10, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 10, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Exporting in ASP.NET MVC Schedule Component

The Scheduler supports exporting all its appointments both to an Excel or ICS extension file at client-side. It offers different client-side methods to export its appointments in an Excel or iCal format file. Let's look onto the ways on how to implement the exporting functionality in Scheduler.

#### Excel Exporting

The Scheduler allows you to export all its events into an Excel format file by using the `[exportToExcel]` client-side method. By default, it exports all the default fields of Scheduler mapped through `eventSettings` property.

**Note:** Before you start with excel exporting functionality, you need to import and inject the `ExcelExport` module from the '@syncfusion/ej2-schedule' package using the `Inject` method of Scheduler.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-excel-export::before {
 content: '\e242';
 }
 .e-schedule-toolbar .e-toolbar-item.e-today {
 display: none !important;
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.exportToExcel();
 }
</script>
```

```
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

#### Exporting with custom fields

By default, Scheduler exports all the default event fields that are mapped to it through the `eventSettings` property. To limit the number of fields on the exported excel file, it provides an option to export only the custom fields of the event data. To export such custom fields alone, define the required fields and pass it as argument to the `exportToExcel` method as shown in the following example. For example: `['Id', 'Subject', 'StartTime', 'EndTime', 'Location']`.



**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-excel-export::before {
 content: '\e242';
 }
 .e-schedule-toolbar .e-toolbar-item.e-today {
 display: none !important;
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var exportValues = {
 fields: ['Id', 'Subject', 'StartTime', 'EndTime', 'Location']
 };
 scheduleObj.exportToExcel(exportValues);
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()

```

```

{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### Exporting individual occurrences of a recurring series

By default, the Scheduler exports recurring events as a single data by exporting only its parent record into the excel file. If you want to export each individual occurrences of a recurring series appointment as separate records in an Excel file, define the `includeOccurrences` option as `true` and pass it as argument to the `exportToExcel` method. By default, the `includeOccurrences` option is set to `false`.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-excel-export::before {
 content: '\e242';
 }

```

```

 }
 .e-schedule-toolbar .e-toolbar-item.e-today {
 display: none !important;
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var exportValues = { includeOccurrences: true };
 scheduleObj.exportToExcel(exportValues);
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Explosion of Betelgeuse Star",
 Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0),
 EndTime = new DateTime(2019, 1, 8, 11, 0, 0),
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=10"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
}

```

```

public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public string RecurrenceRule { get; set; }
}

```

### Exporting custom event data

By default, the whole event collection bound to the Scheduler gets exported as an excel file. To export only specific events of Scheduler or some custom event collection, you need to pass those custom data collection as a parameter to the `exportToExcel` method as shown in this following example, through the `customData` option.

**Note:** By default, the event data are taken from Scheduler dataSource.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-excel-export::before {
 content: '\e242';
 }
 .e-schedule-toolbar .e-toolbar-item.e-today {
 display: none !important;
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var exportValues = {
 customData: [{
 Id: 1,
 Subject: 'Explosion of Betelgeuse Star',
 Location: 'Space Centre USA',
 StartTime: new Date(2019, 0, 6, 9, 30),
 EndTime: new Date(2019, 0, 6, 11, 0),
 }

```

```

 CategoryColor: '#1aaa55'
 }, {
 Id: 2,
 Subject: 'Thule Air Crash Report',
 Location: 'Newyork City',
 StartTime: new Date(2019, 0, 7, 12, 0),
 EndTime: new Date(2019, 0, 7, 14, 0),
 CategoryColor: '#357cd2'
 }]
};
scheduleObj.exportToExcel(exportValues);
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
 };
 ViewBag.view = viewOption;
 return View();
}

public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
}

```

```

public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
}

```

### Export with custom file name

By default, the Scheduler allows you to download the exported Excel file with a name `Schedule.xlsx`. It also provides an option to export the excel file with a custom file name, define the desired `fileName` and passing it as an argument to the `exportToExcel` method.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-excel-export::before {
 content: '\e242';
 }
 .e-schedule-toolbar .e-toolbar-item.e-today {
 display: none !important;
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var exportValues = { fileName: "SchedulerData" };
 scheduleObj.exportToExcel(exportValues);
 }
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
}

```

```

List<ScheduleView> viewOption = new List<ScheduleView>()
{
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
};
ViewBag.view = viewOption;
return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Excel file formats

By default, the Scheduler exports event data to an excel file in the .xlsx format. You can also export the Scheduler data in either of the file type such as .xlsx or csv formats, by defining the `exportType` option as either `csv` or `xlsx`. By default, the `exportType` is set to `xlsx`.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })

```

```

 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-excel-export::before {
 content: '\e242';
 }
 .e-schedule-toolbar .e-toolbar-item.e-today {
 display: none !important;
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var exportValues = { exportType: "csv" };
 scheduleObj.exportToExcel(exportValues);
 }
 }
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData

```



```

 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### Custom separator in CSV

The Scheduler exports the event data to CSV format with `,` as separator. You can change separator by setting `separator` property in `ExportOptions`.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .Views(ViewBag.view)
 .ActionBegin("onActionBegin")
 .EventSettings(new ScheduleEventSettings {
 dataSource: [
 {
 Id: 1,
 Subject: 'Explosion of Betelgeuse Star',
 StartTime: new Date(2022, 0, 8, 9, 30),
 EndTime: new Date(2022, 0, 8, 11, 0),
 Location: 'Chennai',
 OwnerId: 1
 }, {
 Id: 2,
 Subject: 'Thule Air Crash Report',
 StartTime: new Date(2022, 0, 10, 12, 0),
 EndTime: new Date(2022, 0, 10, 14, 0),
 Location: 'Mumbai',
 OwnerId: 2
 }, {
 Id: 3,
 Subject: 'Blue Moon Eclipse',
 StartTime: new Date(2022, 0, 13, 9, 30),
 EndTime: new Date(2022, 0, 13, 11, 0),
 Location: 'Mumbai',
 }
]
 })

```

```

 OwnerId: 3
 },
]
})
.SelectedDate(new DateTime(2022, 2, 15))
.Render()
)
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var exportValues = { exportType: 'csv', separator: ';' };
 scheduleObj.exportToExcel(exportValues);
 }
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2022, 2, 8, 9, 30, 0), EndTime = new DateTime(2022,
 2, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2022, 2, 9, 12, 0, 0), EndTime = new DateTime(2022,
 2, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2022, 2, 10, 10, 30, 0), EndTime = new
 DateTime(2022, 2, 10, 11, 0, 0) });
 appData.Add(new AppointmentData

```

```

 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2022, 2, 11, 13, 0, 0), EndTime = new
 DateTime(2022, 2, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2022, 2, 12, 12, 0, 0), EndTime = new
 DateTime(2022, 2, 12, 14, 0, 0) });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Exporting calendar events as ICS file

You can export the Scheduler events to a calendar (.ics) file format, and open it on any of the other default calendars such as Google or Outlook.

The following code example shows how the Scheduler events are exported to a calendar (.ics) file by making use of the `exportToCalendar` public method.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@Html.EJS().Button("ics-export").Content("Export").Render()
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .calendar-import.e-upload {
 border: 0;
 padding-left: 0 !important;
 }
 .calendar-import.e-upload .e-file-select-wrap {
 padding: 0
 }
 .calendar-import.e-upload .e-file-select-wrap .e-file-drop {
 display: none;
 }
</style>
<script type="text/javascript">
 document.getElementById('ics-export').onclick = function () {
 var scheduleObj =
 document.getElementById('schedule').ej2_instances[0];
 scheduleObj.exportToICalendar();
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

*Customizing column header with custom fields exporting*

Using fields property, we can only export the defined fields into excel without customizing the header. Now we can provide the alternate support to customize the header of custom fields exporting using the `fieldsInfo` option through the `ExportFieldInfo` interface and pass it as an argument to the `exportToExcel` method as shown in the following example.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")

```

```

 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .ActionBegin("onActionBegin")
 .Render()
)
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-excel-export',
 text: 'Excel Export', cssClass: 'e-excel-export', click:
onExportClick
 };
 args.items.push(exportItem);
 }
 function onExportClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var customFields = [
 { name: 'Subject', text: 'Summary' },
 { name: 'StartTime', text: 'First Date' },
 { name: 'EndTime', text: 'Last Date' },
 { name: 'Location', text: 'Place' },
 { name: 'OwnerId', text: 'Owners' }
];
 var exportValues = { fieldsInfo: customFields };
 scheduleObj.exportToExcel(exportValues);
 }
 }
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
}

```

```

 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

#### *Exporting calendar with custom file name*

By default, the calendar is exported with a file name `Calendar.ics`. To change this file name on export, pass the custom string value as `fileName` to the method argument so as to get the file downloaded with this provided name.

The following example downloads the iCal file with a name `ScheduleEvents.ics`.

#### **CSHTML**

```

@using Syncfusion.EJ2.Schedule
@Html.EJS().Button("ics-export").Content("Export").Render()
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .calendar-import.e-upload {
 border: 0;
 padding-left: 0 !important;
 }
 .calendar-import.e-upload .e-file-select-wrap {
 padding: 0
 }
 .calendar-import.e-upload .e-file-select-wrap .e-file-drop {
 display: none;
 }
</style>
<script type="text/javascript">
 document.getElementById('ics-export').onclick = function () {

```

```

var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.exportToICalendar('ScheduleEvents');
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

## Import events from other calendars

The events from external calendars (ICS files) can be imported into Scheduler by using the `importICalendar` method. This method accepts the `blob` object of an .ics file to be imported, as a mandatory argument.

The following example shows how to import an ICS file into Scheduler, using the `importICalendar` method.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
@Html.EJS().Uploader("ics-
import").AllowedExtensions(".ics").CssClass("calendar-
import").Multiple(false).ShowFileList(false).Buttons(new
Syncfusion.EJ2.Inputs.UploaderButtonsProps { Browse = "Choose file"
}).Selected("onSelected").Render()
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2019, 1, 10))
 .Render()
)
<style>
 .calendar-import.e-upload {
 border: 0;
 padding-left: 0 !important;
 }
 .calendar-import.e-upload .e-file-select-wrap {
 padding: 0
 }
 .calendar-import.e-upload .e-file-select-wrap .e-file-drop {
 display: none;
 }
</style>
<script type="text/javascript">
 window.onload = function (args) {
 var uploaderObj = document.getElementById("ics-
import").ej2_instances[0];
 uploaderObj.setProperties({
 buttons: {
 browse: 'Choose file',
 }
 })
 }
 function onSelected(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.importICalendar(args.event.target.files[0]);
 }
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData

```



```

 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
 }
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### How to print the Scheduler element

The Scheduler allows you to print the Scheduler element by using the `print` client-side method. The `print` method works in two ways. You can find it below.

- Using `print` method without options.
- Using a `print` method with options.

#### Using `print` method without options

You can print the Schedule element with the current view by using the `print` method without passing the options. The following example shows how to print the Scheduler using the `print` method without passing options.

#### CSHTML

```

@using Syncfusion.EJ2.Schedule

@Html.EJS().Schedule("schedule").Width("100%").Height("100%").ActionBegin("o
nActionBegin").EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource }).SelectedDate(new DateTime(2019, 1, 10)).Render()

<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-print::before {
 content: '\e813';
 }
</style>

```

```

<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-
schedule-print',
 text: 'Print', cssClass: 'e-print', click: onPrintIconClick
 };
 args.items.push(exportItem);
 }
 }
 function onPrintIconClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.print();
 }
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
}

```

```
public DateTime EndTime { get; set; }
}
```

### Using a print method with options

You can print the Schedule element based on your needs using the `print` method by passing the print options used in this example with its values. The following example shows how to print the Scheduler using the `print` method by passing the options.

### CSHTML

```
@using Syncfusion.EJ2.Schedule

@Html.EJS().Schedule("schedule").Width("100%").Height("100%").ActionBegin("onActionBegin").EventSettings(new ScheduleEventSettings { DataSource = ViewBag.datasource }).SelectedDate(new DateTime(2019, 1, 10)).Render()

<style>
 .e-schedule .e-schedule-toolbar .e-icon-schedule-print::before {
 content: '\e813';
 }
</style>
<script type="text/javascript">
 function onActionBegin(args) {
 if (args.requestType === 'toolbarItemRendering') {
 var exportItem = {
 align: 'Right', showTextOn: 'Both', prefixIcon: 'e-icon-schedule-print',
 text: 'Print', cssClass: 'e-print', click: onPrintIconClick
 };
 args.items.push(exportItem);
 }
 }
 function onPrintIconClick() {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 let printModel = {
 agendaDaysCount: 14,
 cssClass: 'e-print-schedule',
 currentView: scheduleObj.currentView,
 dateFormat: 'dd-MMM-yyyy',
 enableRtl: false,
 endHour: '18:00',
 firstDayOfWeek: 1,
 firstMonthOfYear: 0,
 group: {},
 height: 'auto',
 locale: scheduleObj.locale,
 maxDate: scheduleObj.selectedDate,
 minDate: scheduleObj.getCurrentViewDates()[0],
 readonly: true,
 resources: [],
 rowAutoHeight: false,
 selectedDate: new Date(),
 showHeaderBar: false,
 showTimeIndicator: false,
 showWeekNumber: false,
 showWeekend: false,
```

```

 startHour: '06:00',
 timeFormat: 'HH',
 timeScale: { enable: true },
 width: 'auto',
 workDays: [1, 2, 3, 4, 5],
 workHours: { highlight: true, start: '10:00', end: '20:00' }
 };
 scheduleObj.print(printModel);
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", Location = "Dallas",
 StartTime = new DateTime(2019, 1, 8, 9, 30, 0), EndTime = new DateTime(2019,
 1, 8, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Thule Air Crash Report", Location = "Texas",
 StartTime = new DateTime(2019, 1, 9, 12, 0, 0), EndTime = new DateTime(2019,
 1, 9, 14, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 3, Subject = "Blue Moon Eclipse", Location = "Australia",
 StartTime = new DateTime(2019, 1, 10, 10, 30, 0), EndTime = new
 DateTime(2019, 1, 10, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 4, Subject = "Meteor Showers in 2019", Location = "Canada",
 StartTime = new DateTime(2019, 1, 11, 13, 0, 0), EndTime = new
 DateTime(2019, 1, 11, 14, 30, 0) });
 appData.Add(new AppointmentData
 { Id = 5, Subject = "Milky Way as Melting pot", Location = "Mexico",
 StartTime = new DateTime(2019, 1, 12, 12, 0, 0), EndTime = new
 DateTime(2019, 1, 12, 14, 0, 0) });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public string Location { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## Context menu

You can display context menu on work cells and appointments of Scheduler by making use of the **ContextMenu** control manually from the application end. In the following code example, context menu control is being added from sample end and set its target as **Scheduler**.

On Scheduler cells, you can display the menu items such as **New Event**, **New Recurring Event** and **Today** option. For appointments, you can display its related options such as **Edit Event** and **Delete Event**. The default event window can be opened for appointment creation and editing using the **openEditor** method of Scheduler.

The deletion of appointments can be done by using the **deleteEvent** public method. Also, the **SelectedDate** property can be used to navigate between different dates.

**Note:** You can also display custom menu options on Scheduler cells and appointments. Context menu will open on tap-hold in responsive mode.

## CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("650px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .AllowDragAndDrop(false)
 .AllowResizing(false)
 .Render()
)
@ (Html.EJS().ContextMenu("contextmenu")
 .CssClass("schedule-context-menu")
 .BeforeOpen("onContextMenuBeforeOpen")
 .Select("onMenuItemSelect")
 .Target(".e-schedule")
 .Items(ViewBag.menuItems)
 .Render()
)
<style>
 .schedule-context-menu .e-menu-item .new::before {
 content: '\e7f9';
 }
 .schedule-context-menu .e-menu-item .edit::before {
 content: '\ea9a';
 }
 .schedule-context-menu .e-menu-item .recurrence::before {
 content: '\e308';
 font-weight: bold;
 }
 .schedule-context-menu .e-menu-item .today::before {
 content: '\e322';
 }
 .schedule-context-menu .e-menu-item .delete::before {
 content: '\e94a';
 }
 .e-bigger .schedule-context-menu ul .e-menu-item .e-menu-icon {
```

```

 font-size: 14px;
 }
 .schedule-context-menu ul .e-menu-item .e-menu-icon {
 font-size: 12px;
 }
</style>
<script type="text/javascript">
 var selectedTarget;
 function onContextMenuBeforeOpen(args) {
 var newEventElement = document.querySelector('.e-new-event');
 if (newEventElement) {
 ej.base.remove(newEventElement);
 ej.base.removeClass([document.querySelector('.e-selected-cell')], 'e-selected-cell');
 }
 var scheduleObj = document.querySelector(".e-schedule").ej2_instances[0];
 scheduleObj.closeQuickInfoPopup();
 var targetElement = args.event.target;
 if (ej.base.closest(targetElement, '.e-contextmenu')) {
 return;
 }
 selectedTarget = ej.base.closest(targetElement, '.e-appointment, .e-work-cells, ' +
 '.e-vertical-view .e-date-header-wrap .e-all-day-cells, .e-vertical-view .e-date-header-wrap .e-header-cells');
 if (ej.base.isNullOrUndefined(selectedTarget)) {
 args.cancel = true;
 return;
 }
 if (selectedTarget.classList.contains('e-appointment')) {
 var eventObj = scheduleObj.getEventDetails(selectedTarget);
 if (eventObj.RecurrenceRule) {
 this.showItems(['EditRecurrenceEvent', 'DeleteRecurrenceEvent'], true);
 this.hideItems(['Add', 'AddRecurrence', 'Today', 'Save', 'Delete'], true);
 } else {
 this.showItems(['Save', 'Delete'], true);
 this.hideItems(['Add', 'AddRecurrence', 'Today', 'EditRecurrenceEvent', 'DeleteRecurrenceEvent'], true);
 }
 return;
 }
 this.hideItems(['Save', 'Delete', 'EditRecurrenceEvent', 'DeleteRecurrenceEvent'], true);
 this.showItems(['Add', 'AddRecurrence', 'Today'], true);
 }
 function onMenuItemSelect(args) {
 var scheduleObj = document.querySelector(".e-schedule").ej2_instances[0];
 var selectedMenuItem = args.item.id;
 var eventObj;
 if (selectedTarget.classList.contains('e-appointment')) {
 eventObj = scheduleObj.getEventDetails(selectedTarget);
 }
 switch (selectedMenuItem) {

```

```

 case 'Today':
 scheduleObj.selectedDate = new Date();
 break;
 case 'Add':
 case 'AddRecurrence':
 var selectedCells = scheduleObj.getSelectedElements();
 var activeCellsData =
scheduleObj.getCellDetails(selectedCells.length > 0 ? selectedCells :
selectedTarget);
 if (selectedMenuItem === 'Add') {
 scheduleObj.openEditor(activeCellsData, 'Add');
 } else {
 scheduleObj.openEditor(activeCellsData, 'Add', null, 1);
 }
 break;
 case 'Save':
 case 'EditOccurrence':
 case 'EditSeries':
 if (selectedMenuItem === 'EditSeries') {
 eventObj = new
ej.data.DataManager(scheduleObj.eventsData).executeLocal(new
ej.data.Query().
 where(scheduleObj.eventFields.id, 'equal',
eventObj[scheduleObj.eventFields.recurrenceID])[0];
 }
 scheduleObj.openEditor(eventObj, selectedMenuItem);
 break;
 case 'Delete':
 scheduleObj.deleteEvent(eventObj);
 break;
 case 'DeleteOccurrence':
 case 'DeleteSeries':
 scheduleObj.deleteEvent(eventObj, selectedMenuItem);
 break;
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 List<object> menuItems = new List<object>();
 menuItems.Add(new
 {
 text = "New Event",
 iconCss = "e-icons new",
 id = "Add"
 });
 menuItems.Add(new
 {
 text = "New Recurring Event",
 iconCss = "e-icons recurrence",
 id = "AddRecurrence"
 });
}

```

```

menuItem.Add(new
{
 text = "Today",
 iconCss = "e-icons today",
 id = "Today"
});
menuItem.Add(new
{
 text = "Edit Event",
 iconCss = "e-icons edit",
 id = "Save"
});
menuItem.Add(new
{
 text = "Edit Event",
 id = "EditRecurrenceEvent",
 iconCss = "e-icons edit",
 items = new List<object>() {
 new { text = "Edit Occurrence", id = "EditOccurrence" },
 new { text = "Edit Series", id = "EditSeries" }
 }
});
menuItem.Add(new
{
 text = "Delete Event",
 iconCss = "e-icons delete",
 id = "Delete"
});
menuItem.Add(new
{
 text = "Delete Event",
 id = "DeleteRecurrenceEvent",
 iconCss = "e-icons delete",
 items = new List<object>() {
 new { text = "Delete Occurrence", id = "DeleteOccurrence" },
 new { text = "Delete Series", id = "DeleteSeries" }
 }
});
ViewBag.menuItems = menuItem;
return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 2,
 Subject = "Meeting",
 StartTime = new DateTime(2023, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2023, 2, 15, 12, 30, 0),
 IsAllDay = false,
 Status = "Completed",
 Priority = "High"
 });
 return appData;
}
public class AppointmentData

```



```
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public string Status { get; set; }
 public string Priority { get; set; }
}
```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Scheduler dimensions

The Scheduler dimensions refers to both height and width of the entire layout and it accepts 3 types of values.

- auto
- pixel
- percentage

### Auto Height and Width

When height and width of the Scheduler are set to **auto**, it will try as hard as possible to keep an element the same width as its parent container. In other words, the parent container that holds Scheduler, its width/height will be the sum of its children. By default, Scheduler is assigned with **auto** values for both height and width properties.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("auto")
 .Width("auto")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
```

```

 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

### Height and Width in pixel

The Scheduler height and width will be rendered exactly as per the given pixel values. It accepts both string and number values.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .Width("650px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
}

```

```

public string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime EndTime { get; set; }
public bool IsAllDay { get; set; }
}

```

### Height and Width in percentage

When height and width of the Scheduler are given as percentage, it will make the Scheduler as wide as the parent container.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("100%")
 .Width("100%")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Meeting",
 StartTime = new DateTime(2018, 2, 15, 10, 0, 0),
 EndTime = new DateTime(2018, 2, 15, 12, 30, 0),
 IsAllDay = false,
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## See Also

- [How to Change Scheduler Cell Dimensions](#)

## Recurrence editor

The Recurrence editor is integrated into Scheduler editor window by default, to process the recurrence rule generation for events. Apart from this, it can also be used as an individual component referring from the Scheduler repository to work with the recurrence related processes.

**Note:** All the valid recurrence rule string mentioned in the [iCalendar](#) specifications are applicable to use with the recurrence editor.

## Customizing the repeat type option in editor

By default, there are 5 types of repeat options available in recurrence editor such as,

- Never
- Daily
- Weekly
- Monthly
- Yearly

It is possible to customize the recurrence editor to display only the specific repeat options such as **Daily** and **Weekly** options alone by setting the appropriate **frequencies** option.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .PopupOpen("onPopupOpen")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script type="text/javascript">
 function onPopupOpen(args) {
 if (args.type === 'Editor') {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.eventWindow.recurrenceEditor.frequencies = ['none',
'daily', 'weekly'];
 }
 }
</script>
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
```

```

 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2018, 2, 14, 9, 30, 0), EndTime = new DateTime(2018, 2, 14, 11, 0,
 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

The other properties available in recurrence editor are tabulated below,

| Properties        | Type    | Description                                                                |
|-------------------|---------|----------------------------------------------------------------------------|
| ----- ----- ----- |         |                                                                            |
| FirstDayOfWeek    | number  | Sets the first day of the week on recurrence editor.                       |
| StartDate         | Date    | Sets the start date from which date the recurrence event starts.           |
| DateFormat        | string  | Sets the specific date format on recurrence editor.                        |
| Locale            | string  | Sets the locale to be applied on recurrence editor.                        |
| CssClass          | string  | Allows styling to be applied on recurrence editor with custom class names. |
| EnableRtl         | boolean | Allows recurrence editor to render in RTL mode.                            |
| MinDate           | Date    | Sets the minimum date on recurrence editor.                                |
| MaxDate           | Date    | Sets the maximum date on recurrence editor.                                |
| Value             | string  | Sets the recurrence rule value on recurrence editor.                       |
| SelectedType      | number  | Sets the specific repeat type on the recurrence editor.                    |

#### Customizing the End Type Option in Editor

By default, there are 3 types of end options available in the recurrence editor such as:

- Never
- Until

- Count

It is possible to customize the recurrence editor to display only the specific end options, such as the **Until** and **Count** options alone, by setting the appropriate [endTypes](#) option.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().RecurrenceEditor("RecurrenceEditor").SelectedType(1).Render())
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 var recObject =
document.getElementById('RecurrenceEditor').ej2_instances[0];
 recObject.endTypes = ["until", "count"];
 });
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 return View();
}
```

### Accessing the recurrence rule string

The recurrence rule is usually generated based on the options selected from the recurrence editor and also it follows the [iCalendar](#) specifications. The generated recurrence rule string is a valid one to be used with the Scheduler event's recurrence rule field.

There is a **Change** event available in recurrence editor, that triggers on every time the fields of recurrence editor tends to change. Within this event argument, you can access the generated recurrence value through the **value** option as shown in the following code example.

### CSHTML

```
@using Syncfusion.EJ2.Schedule

<div class="recurrence-editor-wrap">
 <div style="padding-bottom:15px;">
 <label>Rule Output</label>
 <div class="rule-output-container">
 <div id="rule-output">FREQ=DAILY; INTERVAL=1;</div>
 </div>
 </div>

 @(Html.EJS().RecurrenceEditor("RecurrenceEditor").SelectedType(1).Change("onchange").Render())
</div>

<style>
 .recurrence-editor-wrap {
 margin: 0 25%;
 }
 .rule-output-container {
 height: auto;
 }
</style>
```

```

 border: 1px solid #969696;
 }
 #rule-output {
 padding: 8px 4px;
 text-align: center;
 min-height: 20px;
 overflow: hidden;
 overflow-wrap: break-word;
 }
 @@media (max-width: 580px) {
 .recurrence-editor-wrap {
 margin: 0 5%;
 }
 }
</style>
<script type="text/javascript">
 function onChange(args) {
 var outputElement = document.querySelector('#rule-output');
 if(args.value == "") {
 outputElement.innerText = 'Select Rule';
 } else {
 outputElement.innerText = args.value;
 }
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

### Set specific value on recurrence editor

It is possible to display the recurrence editor with specific options loaded initially, based on the rule string that we provide. The fields of recurrence editor will change its values accordingly, when we provide a particular rule through the `setRecurrenceRule` method.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
<div class="recurrence-editor-wrap">
 <div style="padding-bottom:15px;">
 <label>Rule Output</label>
 <div class="rule-output-container">
 <div id="rule-output"></div>
 </div>
 </div>

 @Html.EJS().RecurrenceEditor("RecurrenceEditor").Change("onChange").Render()
</div>

<style>
 .recurrence-editor-wrap {
 margin: 0 25%;
 }

```

```

 }
 .rule-output-container {
 height: auto;
 border: 1px solid #969696;
 }
 #rule-output {
 padding: 8px 4px;
 text-align: center;
 min-height: 20px;
 overflow: hidden;
 overflow-wrap: break-word;
 }
 @@media (max-width: 580px) {
 .recurrence-editor-wrap {
 margin: 0 5%;
 }
 }
</style>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 var recObject =
document.getElementById('RecurrenceEditor').ej2_instances[0];
 recObject.setRecurrenceRule('FREQ=DAILY;INTERVAL=2;COUNT=8');
 });
 function onChange(args) {
 var outputElement = document.querySelector('#rule-output');
 if(args.value == "") {
 outputElement.innerText = 'Select Rule';
 } else {
 outputElement.innerText = args.value;
 }
 }
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

### Recurrence date generation

You can parse the **RecurrenceRule** of an event to generate the date instances on which that particular event is going to occur, using the **getRecurrenceDates** method. It generates the dates based on the **RecurrenceRule** that we provide. The parameters to be provided for **getRecurrenceDates** method are as follows.

| Field name        | Type   | Description                                 |
|-------------------|--------|---------------------------------------------|
| ----- ----- ----- | -----  | -----                                       |
| startDate         | Date   | Appointment start date.                     |
| rule              | String | Recurrence rule present in an event object. |



| **excludeDate** | String | Date collection (in ISO format) to be excluded. It is **optional**. |

| **maximumCount** | Number | Number of date count to be generated. It is **optional**. |

| **viewDate** | Date | Current view range's first date. It is **optional**. |

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
<div>
 <div style="padding-bottom:15px;">
 <label id="rule-label">Rule Output</label>
 <div class="rule-output-container">
 <div id="rule-output"></div>
 </div>
 </div>
 @Html.EJS().RecurrenceEditor("RecurrenceEditor").Render()
</div>
<style>
 .recurrence-editor-wrap {
 margin: 0 25%;
 }
 .rule-output-container {
 height: auto;
 border: 1px solid #969696;
 }
 #rule-output {
 padding: 8px 4px;
 text-align: center;
 min-height: 20px;
 overflow: hidden;
 overflow-wrap: break-word;
 }
 #RecurrenceEditor {
 display: none;
 }
</style>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 var outputElement = document.querySelector('#rule-output');
 var labelElement = document.querySelector('#rule-label');
 var recObject = new ej.schedule.RecurrenceEditor();
 var dates = recObject.getRecurrenceDates(new Date(2018, 0, 7, 10, 0), 'FREQ=DAILY;INTERVAL=1', '20180108T114224Z,20180110T114224Z', 4, new Date(2018, 0, 7))
 labelElement.innerText = 'Date Collections';
 outputElement.innerHTML = '';
 for (var index = 0; index < dates.length; index++) {
 outputElement.appendChild(new ej.base.createElement('div', {
 innerHTML: new Date(dates[index]).toString()
 }));
 }
 });
</script>
```

**DATA.CS**

```
public ActionResult Index()
```

```
{
 return View();
}
```

**Note:** Above example will generate two dates January 7, 2018 & January 9 2018 by excluding the in between dates January 8 2018 & January 10 2018, since those dates were given in the exclusion list. Generated dates can then be utilized to create appointments.

#### Recurrence date generation in server-side

It is also possible to generate recurrence date instances from server-side by manually referring the `RecurrenceHelper` class, which is specifically written and referred from application end to handle this date generation process.

**Note:** Refer [here](#) for the step by step procedure to achieve date generation in server-side.

#### Restrict date generation with specific count

In case, if the rule is given in "NEVER ENDS" category, then you can mention the maximum count when you actually want to stop the date generation starting from the provided start date. To do so, provide the appropriate `maximumCount` value within the `getRecurrenceDates` method as shown in the following code example.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
<div>
 <div style="padding-bottom:15px;">
 <label id="rule-label">Rule Output</label>
 <div class="rule-output-container">
 <div id="rule-output"></div>
 </div>
 </div>
 @Html.EJS().RecurrenceEditor("RecurrenceEditor").Render()
</div>
<style>
 .recurrence-editor-wrap {
 margin: 0 25%;
 }
 .rule-output-container {
 height: auto;
 border: 1px solid #969696;
 }
 #rule-output {
 padding: 8px 4px;
 text-align: center;
 min-height: 20px;
 overflow: hidden;
 overflow-wrap: break-word;
 }
 #RecurrenceEditor {
 display: none;
 }
</style>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 var outputElement = document.querySelector('#rule-output');
```

```

var labelElement = document.querySelector('#rule-label');
var recObject = new ej.schedule.RecurrenceEditor();
var dates = recObject.getRecurrenceDates(new Date(2018, 0, 7, 10, 0), 'FREQ=DAILY;INTERVAL=1; COUNT=30', '20180108T114224Z,20180110T114224Z', 10, new Date(2018, 0, 7));
labelElement.innerText = 'Date Collections';
outputElement.innerHTML = '';
for (var index = 0; index < dates.length; index++) {
 outputElement.appendChild(new ej.base.createElement('div', {
 innerHTML: new Date(dates[index]).toString() }));
}
});
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 return View();
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Globalization and Localization

The Scheduler integrates different date-time formats and cultures, which allows it to function globally, thus meeting the diverse needs of different regions.

You can adapt the Scheduler to various languages by parsing and formatting the date or number ([Internationalization](#)), adding culture specific customization and translation to the text ([Localization](#)).

#### Globalization

The Internationalization library provides support for formatting and parsing the number, date, and time by using the official [Unicode CLDR](#) JSON data and also provides the `loadCldr` method to load the culture specific CLDR JSON data.

By default, Scheduler is set to follow the English culture ('en-US'). If you want to go with different culture other than English, follow the below steps.

Install the `CLDR-Data` package by using the below command (it installs the CLDR JSON data). For more information about CLDR-Data, refer to this [link](#).

,

```
npm install cldr-data --save
```

,

Once the package is installed, you can find the culture specific JSON data under the location `node_modules/cldr-data`.

Once the `CLDR-Data` is installed create a folder `cldr-data` inside the `Scripts` folder. Then create the folder directory like shown below in the structure inside the `Scripts` folder.

- Scripts/cldr-data/supplemental
- Scripts/cldr-data/main

The files named as below are required to setup the specific culture to the Schedule.

- numberingSystems.json
- ca-gregorian.json
- numbers.json
- timeZoneNames.json
- ca-islamic.json

The file named `numberingSystems.json` is available in the location `node_modules/cldr-data/supplemental` which is common for all the cultures. Now you can move this file to the location `Scripts/cldr-data/supplemental`.

The other required files mentioned above are available in the location `node_modules/cldr-data/main/culturecode`. In this location every culture having the culture files inside the folder named as its language culture code. For example if we are loading the German culture we can find the German culture files inside the location `node_modules/cldr-data/main/de`. Now create a folder named `de` inside the location `Scripts/cldr-data/main` and move the files inside it.

Now use the `loadCultureFiles` method to load the culture specific CLDR JSON data.

```
`sh
loadCultureFiles('de');
function loadCultureFiles(name) {
var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
var loader = ej.base.loadCldr;
var loadCulture = function (prop) {
var val, ajax;

ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);

ajax.onSuccess = function (value) {
val = value;
};

ajax.send();
loader(JSON.parse(val));
};

for (var prop = 0; prop < files.length; prop++) {
loadCulture(prop);
}
}
```

```
}
,
```

Set the culture to Scheduler by using the `Locale` property.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Locale("fr-CH")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script>
 loadCultureFiles('fr-CH');
 function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numberingSystems.json',
'numbers.json', 'timeZoneNames.json', 'ca-islamic.json'];
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;
 if (files[prop] === 'numberingSystems.json') {
 ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-
data/supplemental/' + files[prop], 'GET', false);
 } else {
 ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-
data/main/' + name + '/' + files[prop], 'GET', false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
 };
 for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
 }
 }
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
```

```

 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2018, 2, 14, 9, 30, 0), EndTime = new DateTime(2018, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Localizing the static Scheduler text

[Localization](#) library allows to display all the static text, date content, and time mode of the Scheduler following the localized language. To achieve this, set the `Locale` property of Scheduler, as well as define the translation text of static words of Scheduler through the `load` method of `L10n` class.

For example, the following code example lets you to define the Hungarian translation words for all the static words used in Scheduler.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Locale("hu")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
<script>
 var L10n = ej.base.L10n;
 L10n.load({
 "hu": {
 "schedule": {
 "day": "Nap",
 "week": "Hét",
 "workWeek": "Munkahét",
 "month": "Hónap",
 "agenda": "Napirend",
 "weekAgenda": "Hét menetrend",
 "workWeekAgenda": "Munkahét napirend",
 "monthAgenda": "Havi menetrend",
 "today": "Ma",
 "noEvents": "Nincs esemény",
 "emptyContainer": "Ezen a napon nincsenek események.",
 "allDay": "Egész nap",
 "start": "Rajt",
 "end": "vég",
 "more": "több",
 "close": "Bezárás",
 "cancel": "Megszünteti",
 }
 }
 });

```

```

 "noTitle": "(Nincs cím)",
 "delete": "Töröl",
 "deleteEvent": "Esemény törlése",
 "deleteMultipleEvent": "Több esemény törlése",
 "selectedItems": "A kiválasztott elemek",
 "deleteSeries": "Sorozat törlése",
 "edit": "szerkesztése",
 "editSeries": "Szerkesztés",
 "editEvent": "Esemény szerkesztése",
 "createEvent": "teremt",
 "subject": "Tantárgy",
 "addTitle": "Cím hozzáadása",
 "moreDetails": "További részletek",
 "save": "Mentés",
 "editContent": "Csak ezt az eseményt vagy egész sorozatot
szeretné szerkeszteni?",
 "deleteRecurrenceContent": "Csak ezt az eseményt vagy egész
sorozatot szeretné törölni?",
 "deleteContent": "Biztosan törölni szeretné ezt az
eseményt?",
 "deleteMultipleContent": "Biztosan törli a kiválasztott
eseményeket?",
 "newEvent": "Új esemény",
 "title": "Cím",
 "location": "Elhelyezkedés",
 "description": "Leírás",
 "timezone": "Időzóna",
 "startTimezone": "Indítsa el az időzónát",
 "endTimezone": "Időzóna vége",
 "repeat": "Ismétlés",
 "saveButton": "Mentés",
 "cancelButton": "Megszünteti",
 "deleteButton": "Töröl",
 "recurrence": "Ismétlődés",
 "wrongPattern": "Az ismétlődési minta nem érvényes.",
 "seriesChangeAlert": "A sorozat egyes példányaiban
végrehajtott módosítások törlésre kerülnek, és ezek az események ismét
megegyeznek a sorozattal.",
 "createError": "Az esemény időtartamának rövidebbnek kell
lennie, mint a gyakorisága. Rövidítse az időtartamot, vagy változtassa meg
az ismétlődési esemény szerkesztőjének ismétlődési mintáját.",
 "recurrenceDateValidation": "Néhány hónap kevesebb, mint a
kiválasztott dátum. Ezekben a hónapokban az esemény a hónap utolsó napjára
esik.",
 "sameDayAlert": "Ugyanezen esemény két eseménye nem
fordulhat elő ugyanazon a napon.",
 "editRecurrence": "Ismétlés szerkesztése",
 "repeats": "ismétlődés",
 "alert": "Éber",
 "startEndError": "A kiválasztott befejezési dátum a kezdő
dátum előtt történik.",
 "invalidDateError": "A megadott dátumérték érvénytelen.",
 "ok": "Rendben",
 "occurrence": "Esemény",
 "series": "Sorozat",
 "previous": "Előző",
 "next": "Következő",

```

```

 "timelineDay": "Idővonal napja",
 "timelineWeek": "Idősor-hét",
 "timelineWorkWeek": "Idővonal munkahét",
 "timelineMonth": "Idővonal hónap",
 "expandAllDaySection": "kiterjed",
 "collapseAllDaySection": "összeomlás"
 },
 "recurrenceeditor": {
 "none": "Egyik sem",
 "daily": "Napi",
 "weekly": "Heti",
 "monthly": "Havi",
 "month": "Hónap",
 "yearly": "Évi",
 "never": "Soha",
 "until": "Amíg",
 "count": "Számol",
 "first": "Első",
 "second": "Második",
 "third": "Harmadik",
 "fourth": "Negyedik",
 "last": "Utolsó",
 "repeat": "Ismétlés",
 "repeatEvery": "Ismételje meg minden",
 "on": "Ismétlés",
 "end": "vég",
 "onDay": "Nap",
 "days": "Napok)",
 "weeks": "Hét (ok)",
 "months": "Hónap (ok)",
 "years": "Évek)",
 "every": "minden",
 "summaryTimes": "idő (s)",
 "summaryOn": "tovább",
 "summaryUntil": "amíg",
 "summaryRepeat": "ismétlődés",
 "summaryDay": "napok)",
 "summaryWeek": "heti (s)",
 "summaryMonth": "hónap (ok)",
 "summaryYear": "évek)",
 "monthWeek": "Hónap",
 "monthPosition": "Havi pozíció",
 "monthExpander": "Hónaposító",
 "yearExpander": "Év bővítő",
 "repeatInterval": "Ismételje meg az intervallumot"
 },
 "calendar": {
 "today": "Ma"
 }
}
});
loadCultureFiles('hu');
function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numberingSystems.json',
'numbers.json', 'timeZoneNames.json', 'ca-islamic.json'];
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {

```



```

 var val, ajax;
 if (files[prop] === 'numberingSystems.json') {
 ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-
data/supplemental/' + files[prop], 'GET', false);
 } else {
 ajax = new ej.base.Ajax(location.origin + '/../Scripts/cldr-
data/main/' + name + '/' + files[prop], 'GET', false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
 };
 for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
 }
}
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Setting date format

Scheduler can be used with all valid date formats and by default it follows the universal date format "MM/dd/yyyy". If the `DateFormat` property is not specified particularly, then it will work based on the locale that is assigned to the Scheduler. As the default locale applied on Scheduler is "en-US", this makes it to follow the "MM/dd/yyyy" pattern.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule"))

```

```

.Width("100%")
.Height("550px")
.DateFormat("yyyy/MM/dd")
.EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
.SelectedDate(new DateTime(2018, 2, 15))
.Render()
)

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

## Setting the time format

Time formats is a way of representing the time value in different string formats in the Scheduler. By default, the time mode of the Scheduler can be either 12 or 24 hours format which is completely based on the **locale** set to the Scheduler. Since the default **locale** value of the Scheduler is en-US, the time mode will be set to 12 hours format automatically. You can also customize the format by using the **timeFormat** property. To know more about the time format standards, refer to the [Date and Time Format](#) section.

The following example demonstrates the Scheduler component in 24 hours format.

## CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
.Width("100%")
.Height("550px")
.TimeFormat("HH:mm")
.EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
.SelectedDate(new DateTime(2018, 2, 15))
.Render())

```

```
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
 DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
 0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

**Note:** `timeFormat` property only accepts the valid time format's.

**Displaying Scheduler in RTL mode**

The Scheduler layout and its behavior can be changed as per the common RTL (Right to Left) conventions by setting `EnableRtl` to `true`. By doing so, the Scheduler will display its usual layout from right to left. Its default value is `false`.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
 ViewBag.datasource })
 .EnableRtl(true)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
```

```

{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Explosion of Betelgeuse Star", StartTime = new
DateTime(2023, 2, 14, 9, 30, 0), EndTime = new DateTime(2023, 2, 14, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

See Also

- [How to change first day of the week in the Scheduler](#)

### Accessibility in ASP.NET MVC Schedule control

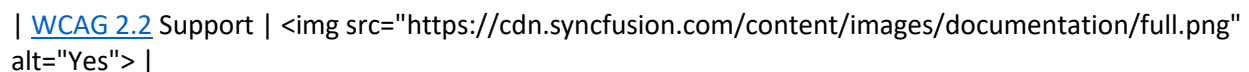
The Scheduler has been designed based on the WAI-ARIA specifications, thus applying the appropriate ARIA roles, states and properties for the Scheduler elements. It is also available with a built-in keyboard navigation support, making it easier for the people who use assistive technologies or who completely rely on the Keyboard support. As per the accessibility standard, the navigated dates, views and other interactive actions performed on the Scheduler will be read out to the target users who use assistive technologies such as screen readers.

The Scheduler makes use of the most required ARIA attributes such as `aria-label` and `role` to support the accessibility in it. To be more accurate, it must be used with an ARIA compliant browser along with the screen reader running from backend.

The accessibility compliance for the Schedule control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" |

| [Section 508](#) Support |  alt="Yes" |

| Screen Reader Support |  alt="Yes" |

| Right-To-Left Support |  alt="Yes" |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

### ARIA attributes

The Scheduler parent element is assigned with a role of **main**, to denote it as the main content of a control as well as a unique element of the entire document.

The following ARIA attributes are used in the Scheduler.

| Attributes | Description |

|-----|-----|

| role="main" | Attribute added to the Scheduler element describes the actual role of the element and denote it as a main and unique content. |

| role="button" | Attribute is assigned to the appointments of Scheduler, to denote it as a clickable element. |

| aria-label | Attribute is set to the Scheduler parent element and its default value is Scheduler's current date. On every time, the date is navigated, this attribute is updated with appropriate current date values. It is also assigned to other scheduler UI elements such as previous and next date navigation buttons depicting its purpose, div element displaying date range in the header bar and appointment elements. |

| aria-labelledby | It indicates editor dialog title to the user through assistive technologies. |

| aria-describedby | It indicates editor dialog content description to the user through assistive technologies. |

| aria-disabled | Attribute is set to the appointment element to indicates the disabled state of the Scheduler.

### Keyboard interaction

All the Scheduler actions can be controlled via keyboard keys by using the `allowKeyboardInteraction` property which is set to `true` by default. The following are the standard keys that work on Scheduler.

| Keys | Description |

|-----|-----|

| Alt + j | Focuses the Scheduler element [provided from application end]. |

| Tab | Focuses the first or active item on the Scheduler header bar and then move the focus to the next available event elements. If no events present, then focus moves out of the control. |

| Shift + Tab | Reverse focusing of the `Tab` key functionality. Inverse focusing of event elements from the last one and then move onto the first or active item on Scheduler header bar and then moves out of the control. |

| Enter | Opens the quick info popup on the selected cells or events. |

| Escape | Closes any of the popup that are in open state. |

| Arrow | To move onto the next available cells in either of the needed directions. (left, right, top and right) |

| Shift + Arrow | For multiple cell selection on either direction. |

| Delete | Deletes one or more selected events. |

| Ctrl + Click on events | To select multiple events. |

| Alt + Number (from 1 to 6) | To switch between the views of Scheduler. |

| Ctrl + Left Arrow | To navigate to the previous date period. |

| Ctrl + Right Arrow | To navigate to the next date period. |

| Left or Right Arrow | On pressing any of these keys, when focus is currently on the Schedule header bar, moves the focus to the previous or next items in the header bar. |

| Space or Enter | It activates any of the focused items. |

| Page Up & Page Down | To scroll through the work cells area. |

| Home | To move the selection to the first cell of Scheduler. |

N> You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

### Ensuring accessibility

The Scheduler control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Scheduler control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Scheduler control with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC control](#)

## Styling and appearance

To modify the Scheduler appearance, you need to override the default CSS of Scheduler. Also, you have an option to create your own custom theme using our [Theme Studio](#). Find the list of CSS classes in Scheduler.

| Css class                                                                                | Purpose                                                      |
|------------------------------------------------------------------------------------------|--------------------------------------------------------------|
| ----- -----                                                                              |                                                              |
| .e-schedule .e-vertical-view .e-work-cells                                               | Work cells in vertical views of scheduler                    |
| .e-schedule .e-month-view .e-work-cells                                                  | Work cells in month view of scheduler                        |
| .e-schedule .e-month-view .e-other-month                                                 | Work cells of other month in month view of scheduler         |
| .e-schedule .e-timeline-view .e-work-cells                                               | Work cells in timeline views of scheduler                    |
| .e-schedule .e-timeline-month-view .e-work-cells                                         | Work cells in timeline month view of scheduler               |
| .e-schedule .e-timeline-year-view .e-work-cells                                          | Work cells in timeline year view of scheduler                |
| .e-schedule .e-timeline-year-view .e-work-cells.e-other-month                            | Work cells of other month in timeline year view of scheduler |
| .e-schedule .e-month-agenda-view .e-work-cells                                           | Work cells in month agenda view of scheduler                 |
| .e-schedule .e-month-agenda-view .e-other-month                                          | Work cells of other month in month agenda view of scheduler  |
| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-other-month | Work cells of other month in year view of scheduler          |
| .e-schedule .e-vertical-view .e-all-day-cells                                            | All day cells in vertical views of scheduler                 |
| .e-schedule .e-vertical-view .e-work-hours                                               | Work hour cells in vertical views of scheduler               |
| .e-schedule .e-month-view .e-work-days                                                   | Work day cells in month view of scheduler                    |
| .e-schedule .e-month-agenda-view .e-work-days                                            | Work day cells in month agenda view of scheduler             |
| .e-schedule .e-timeline-view .e-work-hours                                               | Work hour cells in timeline views of scheduler               |
| .e-schedule .e-timeline-month-view .e-work-days                                          | Work day cells in timeline month view of scheduler           |
|                                                                                          |                                                              |
| .e-schedule .e-timeline-year-view .e-work-cells.e-work-days                              | Work day cells in timeline year view of scheduler            |
| .e-schedule .e-vertical-view .e-day-wrapper .e-appointment                               | Appointment in vertical views of scheduler                   |
| .e-schedule .e-vertical-view .e-all-day-appointment-wrapper .e-appointment               | All day Appointment in vertical views of scheduler           |

| .e-schedule .e-month-view .e-appointment | Appointment in month view of scheduler |

| .e-schedule .e-timeline-view .e-appointment | Appointment in timeline views of scheduler |

| .e-schedule .e-timeline-month-view .e-appointment | Appointment in timeline month view of scheduler |

| .e-schedule .e-timeline-year-view .e-event-table .e-appointment | Appointment in timeline year view of scheduler |

| .e-schedule .e-year-view .e-calendar-wrapper .e-month-calendar.e-calendar .e-appointment | Appointment in year view of scheduler |

| .e-schedule .e-agenda-view .e-appointment | Appointment in agenda view of scheduler |

| .e-schedule .e-month-agenda-view .e-appointment-indicator | Appointment in month agenda view of scheduler |

| .e-schedule .e-block-appointment | Block appointment in scheduler |

| .e-schedule .e-read-only | Read only appointment in scheduler. |

| e-appointment-border | Appointment which are currently selected, use the appointment class hierarchical based on your views. |

| e-selected-cells | work cells which are currently selected, use the work cell class hierarchical based on your views. |

| e-header-cells | Header cells of scheduler, use the work cells hierarchical based on your views. |

| .e-schedule .e-vertical-view .e-resource-cells| Resource cells in vertical views of scheduler. |

| .e-schedule .e-month-view .e-resource-cells| Resource cells in month view of scheduler. |

| .e-schedule .e-timeline-view .e-resource-cells | Resource cells in timeline views of scheduler. |

| .e-schedule .e-timeline-month-view .e-resource-cells| Resource cells in timeline month view of scheduler. |

| e-parent-node | Parent resource cells in timeline views of scheduler. |

| e-child-node | Child resource cells in timeline views of scheduler. |

**Note:** You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## Migration from Essential JS 1

This topic shows the API equivalent of JS2 Scheduler component to be used, while migrating your project that uses JS1 Scheduler.

### Scheduler

#### Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To change the display of days count in agenda view | **Property:** *DaysInAgenda* <br/> <br/>  
@ (Html.EJ().Schedule("schedule").CurrentView(CurrentView.Agenda)<br>.AgendaViewSettings(



```

agendaViewSettings => agendaViewSettings.DaysInAgenda(5)) | Property: AgendaDaysCount

@Html.EJS().Schedule("schedule").CurrentView(View.Agenda).AgendaDaysCount(5).Render() |
| Preventing deletion of appointment | Property: AllowDelete

@Html.EJ().Schedule("schedule").AllowDelete(true)) | Not applicable |
| Allows dragging and dropping of appointments | Property: AllowDragAndDrop

@Html.EJ().Schedule("schedule").AllowDragAndDrop(false)) | Property: AllowDragAndDrop

 @Html.EJS().Schedule("schedule").AllowDragAndDrop(false).Render() |
| Enable inline editing of appointments | Property: AllowInline

@Html.EJ().Schedule("schedule").AllowInline(true)) | Not applicable |
| Allow keyboard interactions | Property: AllowKeyboardNavigation

@Html.EJ().Schedule("schedule").AllowKeyboardNavigation(false)) | Property:
AllowKeyboardInteraction

@Html.EJS().Schedule("schedule").AllowKeyboardInteraction(false).Render() |
| Enable resizing of appointments | Property: EnableAppointmentResize

@Html.EJ().Schedule("schedule").EnableAppointmentResize(false)) | Property: AllowResizing

 @Html.EJS().Schedule("schedule").AllowResizing(false).Render() |
| Blocking time intervals | Property: BlockoutSettings

@Html.EJ().Schedule("schedule").BlockoutSettings(fields =>
fields.Enable(true)
.Datasource(blockData)
.Id("Id")
.Subject("Subject")
.StartTime
e
("StartTime")
.EndTime("EndTime")
.IsBlockAppointment("IsBlockAppointment"))
| Property: IsBlock

 public class BlockData
{
public int Id { get; set; }
public
string Subject { get; set; }
public DateTime StartTime { get; set; }
public DateTime
EndTime { get; set; }
public bool IsBlock { get; set; }
}
 |
| Categorizing the appointments | Property: CategorizeSettings

@Html.EJ().Schedule("schedule")
.CategorizeSettings(fields =>
fields.Datasource(CategorizeValue).Enable(true).AllowMultiple(true).Id("Id").Text("Text").Color
("Color").FontColor("FontColor")) | Not applicable |
| Setting cell height | Property: CellHeight

@Html.EJ().Schedule("schedule").CellHeight("40px")) | Not applicable |
| Cell template | Property: WorkCellsTemplateId

@Html.EJ().Schedule("schedule").WorkCellsTemplateId(#workCellTemplate).AllDayCellsTempl
atId("#allDayCellTemplate")) | Property: CellTemplate

@Html.EJS().Schedule("schedule").CellTemplate("#cellTemplate").Render() |
| Setting cell width | Property: CellWidth

@Html.EJ().Schedule("schedule").CellWidth("35px")) | Not applicable |
| CSS class | Property: CssClass

@Html.EJ().Schedule("schedule").CssClass("customstyle")) | Property: CssClass

@Html.EJS().Schedule("schedule").CssClass("customstyle").Render() |

```

| Enabling Context-menu option | **Property:** *e-context-menu-settings* <br/> <br/>  
 @(Html.EJ().Schedule("schedule")<br/>.ContextMenuSettings(contextMenu =>  
 contextMenu.Enable(true).MenuItems(items =>  
 items.Appointment(AppMenu).Cells(CellMenu))) | Not applicable |

| Current view | **Property:** *CurrentView* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").CurrentView(CurrentView.Day)) | **Property:** *CurrentView*  
 <br/><br/> @Html.EJS().Schedule("schedule").CurrentView(View.Day).Render() |

| Date format | **Property:** *DateFormat* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").DateFormat("yyyy/MM/dd")) | **Property:** *DateFormat*  
 <br/><br/> @Html.EJS().Schedule("schedule").DateFormat("yyyy/MM/dd")) |

| Date header template | **Property:** *date-header-template-id* <br/> <br/>  
 @Html.EJS().Schedule("schedule").DateHeaderTemplateId("#dateTemplate")<br/> | **Property:**  
*DateHeaderTemplate* <br/><br/>  
 @Html.EJS().Schedule("schedule").DateHeaderTemplate("#template") |

| Editor template | Not Applicable | **Property:** *EditorTemplate* <br/><br/>  
 @Html.EJS().Schedule("schedule").EditorTemplate("#EditorTemplate").Render() |

| Enable load on demand | **Property:** *EnableLoadOnDemand* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").EnableLoadOnDemand(true)) | Not applicable |

| Enable persistence | **Property:** *EnablePersistence* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").EnablePersistence(false)) | **Property:** *EnablePersistence*  
 <br/><br/> @Html.EJS().Schedule("schedule").EnablePersistence(true).Render() |

| Enable RTL | **Property:** *EnableRTL* <br/> <br/> @Html.EJ().Schedule("schedule").EnableRTL(true))  
 | **Property:** *EnableRTL* <br/><br/> @Html.EJS().Schedule("schedule").EnableRTL(true).Render() |

| Setting end hour of the scheduler | **Property:** *EndHour* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").EndHour(18)) | **Property:** *EndHour* <br/><br/>  
 @Html.EJS().Schedule("schedule").EndHour("20:00").Render() |

| Setting first day of the week | **Property:** *FirstDayOfWeek* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").FirstDayOfWeek(DayOfWeek.Monday)) | **Property:**  
*FirstDayOfWeek* <br/><br/> @Html.EJS().Schedule("schedule").FirstDayOfWeek(1).Render() |

| Height of the scheduler | **Property:** *Height* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").Height("550px")) | **Property:** *Height* <br/><br/>  
 @Html.EJS().Schedule("schedule").Height("550px").Render() |

| Locale | **Property:** *Locale* <br/> <br/> @Html.EJ().Schedule("schedule").Locale("fr-FR")) |  
**Property:** *Locale* <br/><br/> @Html.EJS().Schedule("schedule").Locale("fr-FR").Render() |

| Priority settings for appointments | **Property:** *PrioritySettings* <br/> <br/>  
 @(Html.EJ().Schedule("schedule")<br/>.PrioritySettings(fields =>  
 fields.Datasource(PriorityValue).Enable(true).Text("Text").Value("Value")) | Not applicable |

| Read only | **Property:** *ReadOnly* <br/> <br/> @Html.EJ().Schedule("schedule").ReadOnly(true)) |  
**Property:** *Readonly* <br/><br/> @Html.EJS().Schedule("schedule").Readonly(true).Render() |

| Reminder settings | **Property:** *e-reminder-settings* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").<br/>.ReminderSettings(rem =>  
 rem.Enable(true).AlertBefore(10)) | Not applicable |

| Resource header template | **Property:** *ResourceHeaderTemplateId* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").Group(gr => {gr.Resources(Group);  
 }).ResourceHeaderTemplateId("#resTemplate")<br/>.Resources(res => {  
 res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).ResourceSettings(fields  
 => fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("GroupId")).Add();})) |  
**Property:** *ResourceHeaderTemplate* <br/><br/>  
 @Html.EJS().Schedule("schedule").ResourceHeaderTemplate("#resourceTemplate").Group(gro  
 up => group.Resources(ViewBag.Resources)).Resources(res => {  
 res.DataSource(ViewBag.Doctors).Field("DoctorId").Title("Doctor  
 Name").Name("Doctors").TextField("text").IdField("id").ColorField("color") |

| Current date of the scheduler | **Property:** *CurrentDate* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").CurrentDate(new DateTime(2018, 6, 12))) | **Property:**  
*SelectedDate* <br/><br/> @Html.EJS().Schedule("schedule").SelectedDate(new  
 DateTime(2018,1,31)).Render() |

| Show all day row | **Property:** *ShowAllDayRow* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowAllDayRow(false)) | Not applicable |

| Show appointment navigator | **Property:** *ShowAppointmentNavigator* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowAppointmentNavigator(false)) | Not applicable |

| Show delete confirmation dialog | **Property:** *ShowDeleteConfirmationDialog* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowDeleteConfirmationDialog(false)) | Not applicable |

| Show header bar | **Property:** *ShowHeaderBar* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowHeaderBar(false)) | **Property:** *ShowHeaderBar* <br/><br/>  
 @Html.EJS().Schedule("schedule").ShowHeaderBar(false).Render() |

| Show location field in event window | **Property:** *ShowLocationField* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowLocationField(false)) | Not applicable |

| Show time zone fields in event window | **Property:** *ShowTimeZoneFields* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowTimeZoneFields(false)) | Not applicable |

| Show previous and next month dates in month view | **Property:** *ShowNextPrevMonth* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowNextPrevMonth(false)) | Not applicable |

| Show overflow button | **Property:** *ShowOverflowButton* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowOverflowButton(false)) | **Property:** *RowAutoHeight*  
 <br/><br/> @Html.EJS().Schedule("schedule").RowAutoHeight(true).Render() |

| Show quick popup | **Property:** *ShowQuickWindow* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowQuickWindow(false)) | **Property:** *ShowQuickInfo*  
 <br/><br/> @Html.EJS().Schedule("schedule").ShowQuickInfo(false).Render() |

| Show current time indicator | **Property:** *ShowCurrentTimeIndicator* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").ShowCurrentTimeIndicator(false)) | **Property:**

*ShowTimeIndicator* <br/><br/>

@Html.EJS().Schedule("schedule").ShowTimeIndicator(false).Render() |

| Show week number | Not Applicable | **Property:** ShowWeekNumber <br/><br/>

@Html.EJS().Schedule("schedule").ShowWeekNumber(true).Render() |

| Show weekend days | **Property:** ShowWeekend <br/> <br/>

@(Html.EJ().Schedule("schedule").ShowWeekend(false)) | **Property:** ShowWeekend <br/><br/>

@Html.EJS().Schedule("schedule").ShowWeekend(false).Render() |

| Setting start hour of the scheduler | **Property:** StartHour <br/> <br/>

@(Html.EJ().Schedule("schedule").StartHour(7)) | **Property:** StartHour <br/><br/>

@Html.EJS().Schedule("schedule").StartHour("09:00").Render() |

| Setting time mode on scheduler | **Property:** TimeMode <br/> <br/>

@(Html.EJ().Schedule("schedule").TimeMode(TimeMode.Hour24)) | Not applicable |

| Setting timezone for scheduler | **Property:** TimeZone <br/> <br/>

@(Html.EJ().Schedule("schedule").TimeZone("UTC +05:30")) | **Property:** Timezone <br/><br/>

@Html.EJS().Schedule("schedule").Timezone("UTC").Render() |

| Views in scheduler | **Property:** Views <br/> <br/> @({List<string> views = new List<string>() {

"Day", "Week", "WorkWeek", "Month" });<br> @(Html.EJ().Schedule("schedule").Views(views)

| **Property:** Views <br/><br/> @Html.EJS().Schedule("schedule").Views(ViewBag.view).Render() |

| Width of the scheduler | **Property:** Width <br/> <br/>

@(Html.EJ().Schedule("schedule").Width("100%")) | **Property:** Width <br/><br/>

@Html.EJS().Schedule("schedule").Width("100%").Render() |

| Working days | **Property:** WorkWeek <br/> <br/> @({List<string> workWeek = new List<string>() {

"Monday", "Friday", "Saturday"

});<br> @(Html.EJ().Schedule("schedule").WorkWeek(workWeek) | **Property:** WorkDays

<br/><br/> @({var workDays = new int[] { 1, 3, 5

});<br> @Html.EJS().Schedule("schedule").WorkDays(ViewBag.workDays).Render() |

| Working hours | **Property:** WorkHours <br/> <br/>

@(Html.EJ().Schedule("schedule").WorkHours(hrs => hrs.Highlight(true).Start(8).End(16)) |

**Property:** WorkHours <br/><br/> @Html.EJS().Schedule("schedule").WorkHours(new

ScheduleWorkHours { Highlight = true, Start = "08:00", End = "20:00" }).Render() |

## Resources

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To define resource datasource | **Property:** Resources <br/> @({<br>List<ScheduleFields> Appoint =

new List<ScheduleFields>();<br>Appoint.Add(new ScheduleFields { Id = "1", Subject =

"Meeting", StartTime = new DateTime(2015, 11, 10, 10, 00, 00), EndTime = new

DateTime(2015, 11, 10, 11, 00, 00), Description = "", AllDay = false, Recurrence = false,

RecurrenceRule = "", RoomId = "1", OwnerId = "5" });<br>List<String> Group = new

List<String>();<br>Group.Add("Rooms");<br>Group.Add("Owners");<br>List<ResourceFields>

Room = new List<ResourceFields>();<br>Room.Add(new ResourceFields { Id = "1", Text =

```

"Room1", Color = "#f8a398", GroupId = "1" });
Room.Add(new ResourceFields { Id = "2",
Text = "Room2", Color = "#56ca95", GroupId = "1" });
List<ResourceFields> Owner = new
List<ResourceFields>();
Owner.Add(new ResourceFields { Text = "Nancy", Id = "1", GroupId =
"1", Color = "#ffaa00", Start = 10, End = 18, CustomDays = new List<string> { "monday",
"wednesday", "friday" } });
Owner.Add(new ResourceFields { Text = "Steven", Id = "3",
GroupId = "2", Color = "#f8a398", Start = 6, End = 10, CustomDays = new List<string> {
"tuesday", "thursday" } });
}

@(Html.EJ().Schedule("schedule").Group(group =>
group.Resources(Group)).Resources(res => { res.ResourceSettings(flds =>
flds.Datasource(Owner)).Field("OwnerId").Title("Owner").Name("Owners").Text("text").Id("id")
.Color("color").Add(); }).AppointmentSettings(fields => fields.Datasource(Appoint))) | Property:
Resources
@{
List<Projects> projects = new List<Projects>();
projects.Add(new
Projects { text = "PROJECT 1", id = 1, color = "#cb6bb2" });
projects.Add(new Projects { text
= "PROJECT 2", id = 2, color = "#56ca85" });
List<Categories> categories = new
List<Categories>();
categories.Add(new Categories { text = "Development", id = 1, color =
"#1aaa55" });
categories.Add(new Categories { text = "Testing", id = 2, color = "#7fa900" });

List<String> Resources = new
List<String>();
Resources.Add("Projects");
Resources.Add("Categories");
}

@Html.EJS().Schedule("schedule").Group(group => group.Resources(Resources)).Resources(res
=> {
res.DataSource(projects).Field("ProjectId").Title("Project").Name("Projects").TextField("text").I
dField("id").ColorField("color").Add();
res.DataSource(categories).Field("CategoryId").Title("Category").Name("Categories").TextField(
"text").IdField("id").ColorField("color").Add(); }).Render() |

```

```

| Allowing multiple selection of resources in event window | Property: AllowMultiple

@Html.EJ().Schedule("schedule").Group(gr => { gr.Resources(Group); }).Resources(res =>
{res.Field("RoomId").Title("Room").Name("Rooms").AllowMultiple(true).ResourceSettings(field
s => fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("GroupId")).Add();
res.Field("OwnerId").Title("Owner").Name("Owners").AllowMultiple(true).ResourceSettings(fiel
ds =>
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color").GroupId("GroupId").Start("Start")
.End("End").WorkWeek("WorkWeek")).Add();}).AppointmentSettings(fields =>
fields.Datasource(Appoint)))
 | Property: AllowMultiple

@Html.EJS().Schedule("schedule").Group(group => group.Resources(Resources)).Resources(res
=> {
res.DataSource(projects).Field("ProjectId").Title("Project").Name("Projects").TextField("text").I
dField("id").ColorField("color").Add();
res.DataSource(categories).Field("CategoryId").Title("Category").Name("Categories").TextField(
"text").IdField("id").ColorField("color").AllowMultiple(true).Add(); }).Render() |

```

```

| Setting different work hours for each resource |
@Html.EJ().Schedule("schedule").Group(gr
=> { gr.Resources(Group); }).Resources(res =>
{res.Field("RoomId").Title("Room").Name("Rooms").ResourceSettings(fields =>
fields.Datasource(Room).Text("Text").Id("Id").Color("Color").GroupId("GroupId")).Add();
res.Field("OwnerId").Title("Owner").Name("Owners").ResourceSettings(fields =>

```

```
fields.Datasource(Owner).Text("Text").Id("Id").Color("Color").GroupId("GroupId").Start("Start")
.End("End").WorkWeek("WorkWeek")).Add();}).AppointmentSettings(fields =>
fields.Datasource(Appoint).Id("Id").Subject("Subject").StartTime("StartTime").EndTime("EndTi
me").Description("Description").AllDay("AllDay").Recurrence("Recurrence").RecurrenceRule("R
ecurrenceRule").ResourceFields("RoomId,OwnerId")) | @
List<DoctorRes> doctors = new
List<DoctorRes>();
doctors.Add(new DoctorRes { text = "Will Smith", id = 1, color =
"#ea7a57", workDays = new List<int> { 1, 2, 4, 5 }, startHour = "08:00", endHour = "15:00"
});
doctors.Add(new DoctorRes { text = "Alice", id = 2, color = "rgb(53, 124, 210)", workDays
= new List<int> { 1, 3, 5 }, startHour = "08:00", endHour = "17:00" });
doctors.Add(new
DoctorRes { text = "Robson", id = 3, color = "#7fa900", startHour = "08:00", endHour = "16:00"
});
List<String> Resources = new
List<String>();
Resources.Add("Doctors");

@Html.EJS().Schedule("schedule").Gr
oup(group => group.Resources(Resources)).Resources(res => { res.DataSource(doctors).Add();
}).Views(view => { view.Option(View.WorkWeek).Add(); view.Option(View.Month).Add();
}).Render() |
```

### Group

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To group the resources in scheduler layout | **Property:** *Resources* <br/> @<br>List<Projects> projects = new List<Projects>(); <br> projects.Add(new Projects { text = "PROJECT 1", id = 1, color = "#cb6bb2" }); <br> projects.Add(new Projects { text = "PROJECT 2", id = 2, color = "#56ca85" }); <br> List<Categories> categories = new List<Categories>(); <br> categories.Add(new Categories { text = "Development", id = 1, color = "#1aaa55" }); <br> categories.Add(new Categories { text = "Testing", id = 2, color = "#7fa900" }); <br> var resources = new string[] { "Projects", "Categories" }; <br> } <br/> @(<Html.EJ().Schedule("schedule").Group(group => group.Resources(resources)) | **Property:** *Resources* <br/> @<br>List<Projects> projects = new List<Projects>(); <br> projects.Add(new Projects { text = "PROJECT 1", id = 1, color = "#cb6bb2" }); <br> projects.Add(new Projects { text = "PROJECT 2", id = 2, color = "#56ca85" }); <br> List<Categories> categories = new List<Categories>(); <br> categories.Add(new Categories { text = "Development", id = 1, color = "#1aaa55" }); <br> categories.Add(new Categories { text = "Testing", id = 2, color = "#7fa900" }); <br> var resources = new string[] { "Projects", "Categories" }; <br> } <br/> @Html.EJS().Schedule("schedule").Group(group => group.Resources(resources)).Render() |

| Allow group editing | **Property:** *AllowGroupEdit* <br/> <br/>

```
@(<Html.EJ().Schedule("schedule").Group(group =>
group.Resources(resources).AllowGroupEdit(true)) | Property: AllowGroupEdit

@Html.EJS().Schedule("schedule").Group(group =>
group.Resources(resources).AllowGroupEdit(true)).Render() |
```

| Grouping resources by date | Not applicable | **Property:** *ByDate* <br/><br/>

```
@Html.EJS().Schedule("schedule").Group(group =>
group.Resources(resources).ByDate(true)).Render() |
```

| Grouping resources based on its group ID | Not applicable | **Property:** *ByGroupId* <br/><br/>  
 @Html.EJS().Schedule("schedule").Group(group =>  
 group.Resources(resources).ByGroupId(false)).Render() |

| Enabling compact view on mobile mode | Not applicable | **Property:** *EnableCompactView* <br/><br/>  
 @Html.EJS().Schedule("schedule").Group(group =>  
 group.Resources(resources).EnableCompactView(false)).Render() |

| Header tooltip template | Not applicable | **Property:** *HeaderTooltipTemplate* <br/><br/>  
 @Html.EJS().Schedule("schedule").Group(group =>  
 group.Resources(resources).HeaderTooltipTemplate("#resourceHeader")).Render() |

#### Header Rows

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Adding custom rows in the header in timeline views | Not applicable | **Property:** *HeaderRows*  
 <br/><br/> @Html.EJS().Schedule("schedule").HeaderRows(headerRow => {  
 headerRow.Option(HeaderRowType.Month).Template("#month-template").Add();  
 headerRow.Option(HeaderRowType.Week).Template("#week-template").Add();  
 headerRow.Option(HeaderRowType.Date).Add(); }).Render() |

#### TimeScale

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Enabling time scale | **Property:** *Enable* <br/> <br/> @Html.EJS().Schedule("schedule").TimeScale(ts  
 => ts.Enable(false))) | **Property:** *Enable* <br/><br/>  
 @Html.EJS().Schedule("schedule").TimeScale(ts => ts.Enable(false)).Render() |

| Setting major interval on time scale | **Property:** *majorSlot* <br/> <br/>  
 @Html.EJS().Schedule("schedule").TimeScale(ts => ts.majorSlot(60))) | **Property:** *Interval*  
 <br/><br/> @Html.EJS().Schedule("schedule").TimeScale(ts => ts.Interval(60)).Render() |

| Setting slot count on time scale | **Property:** *minorSlotCount* <br/> <br/>  
 @Html.EJS().Schedule("schedule").TimeScale(ts => ts.majorSlot(60).minorSlotCount(6))) |  
**Property:** *SlotCount* <br/><br/> @Html.EJS().Schedule("schedule").TimeScale(ts =>  
 ts.Interval(60).SlotCount(6)).Render() |

| Defining major slot template | **Property:** *majorSlotTemplateId* <br/> <br/>  
 @Html.EJS().Schedule("schedule").TimeScale(ts =>  
 ts.majorSlot(60).minorSlotCount(6).majorSlotTemplateId("#majorSlotTemplate"))) | **Property:**  
*MajorSlotTemplate* <br/><br/> @Html.EJS().Schedule("schedule").TimeScale(ts =>  
 ts.Interval(60).SlotCount(6).MajorSlotTemplate("#majorSlotTemplate")).Render() |

| Defining minor slot template | **Property:** *minorSlotTemplateId* <br/> <br/>  
 @Html.EJS().Schedule("schedule").TimeScale(ts =>  
 ts.majorSlot(60).minorSlotCount(6).minorSlotTemplateId("#minorSlotTemplate"))) | **Property:**  
*MinorSlotTemplate* <br/><br/> @Html.EJS().Schedule("schedule").TimeScale(ts =>  
 ts.Interval(60).SlotCount(6).MinorSlotTemplate("#minorSlotTemplate")).Render() |



*Quick info templates*

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Template for quick popup | Not applicable | **Property:** *QuickInfoTemplates* <br/><br/>

```
@Html.EJS().Schedule("schedule").QuickInfoTemplates(quickTemplate =>
quickTemplate.Header("#headertemplate").Content("#contentTemplate").Footer("#footerTemplate").Render() |
```

*Event settings*

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Datasource for events | **Property:** *DataSource* <br/> <br/>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(app =>
app.DataSource((IEnumerable)ViewBag.datasource))) | Property: DataSource

@Html.EJS().Schedule("schedule").EventSettings(eve =>
eve.DataSource(ViewBag.datasource)).Render() |
```

| Appointment fields | **Property:** *AppointmentSettings* <br/> <br/>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(app =>
app.Id("Id").Subject("Subject").Location("Location").AllDay("AllDay").Description("Description")
.StartTime("StartTime").EndTime("EndTime").Recurrence("Recurrence").RecurrenceRule("RecurrenceRule"))) | Property: Fields

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.Fields(f => f.Id("Id").Subject(sub => sub.Name("Subject")).Location(loc =>
loc.Name("Location")).IsAllDay(allday => allday.Name("IsAllDay")).Description(desc =>
desc.Name("Description")).StartTime(st => st.Name("StartTime")).EndTime(et =>
et.Name("EndTime")).RecurrenceID(recId =>
recId.Name("RecurrenceID")).RecurrenceRule(recRule =>
recRule.Name("RecurrenceRule")).RecurrenceException(recEx =>
recEx.Name("RecurrenceException")).StartTImezone(stz =>
stz.Name("StartTImezone")).EndTImezone(etz => etz.Name("EndTImezone")))).Render() |
```

| Enabling tooltip for appointments | **Property:** *Enable* <br/> <br/>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(app =>
app.DataSource(ViewBag.datasource).TooltipSettings(pre => pre.Enable(true)))
Note: Here
tooltip setting for events is maintained separately | Property: EnableTooltip

@Html.EJS().Schedule("schedule").EventSettings(eve =>
eve.DataSource(ViewBag.datasource).EnableTooltip(true)).Render() |
```

| Tooltip template for appointments | **Property:** *templateId* <br/> <br/>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(app =>
app.DataSource(ViewBag.datasource).TooltipSettings(pre => pre.templateId("#tooltip"))) |
Property: TooltipTemplate

@Html.EJS().Schedule("schedule").EventSettings(eve =>
eve.DataSource(ViewBag.datasource).EnableTooltip(true).TooltipTemplate("#toolTip")).Render() |
```



| Template for appointments | **Property:** *AppointmentTemplateId* <br/> <br/>  
 @(Html.EJ().Schedule("schedule").AppointmentTemplateId("#appTemplate").AppointmentSettings(app => app.DataSource(ViewBag.datasource)))<br/>Note: Here appointment template is used as simple API | **Property:** *Template* <br/><br/>  
 @Html.EJS().Schedule("schedule").EventSettings(eve => eve.DataSource(ViewBag.datasource).Template("#eventTemplate")).Render() |  
 | Query | **Property:** *query* <br/> <br/> var query = ej.Query().from("Events").take(10);<br/><br/>ej-schedule id="schedule"><br/><e-appointment-settings datasource="Appoint" query="query"></e-appointment-settings><br/></ej-schedule> | **Property:** *Query* <br/><br/>  
 @Html.EJS().Schedule("schedule").EventSettings(eve => eve.DataSource(ViewBag.datasource).Query(ViewBag.query)).Render() |  
 | Define which resource level's color to be applied to events | Not applicable | **Property:** *ResourceColorField* <br/><br/> @Html.EJS().Schedule("schedule").Group(group => group.Resources(ViewBag.Resources)).Resources(res => { res.DataSource(ViewBag.Projects).Field("ProjectId").Title("Project").Name("Projects").TextField("text").IdField("id").ColorField("color").Add(); res.DataSource(ViewBag.Categories).Field("CategoryId").Title("Category").Name("Categories").TextField("text").IdField("id").ColorField("color").AllowMultiple(true).Add();}).EventSettings(e => e.ResourceColorField("Projects").DataSource(ViewBag.datasource)).Render() |

## Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| To add appointments manually | **Method:** *saveAppointment()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields => fields.DataSource((IEnumerable)ViewBag.data))) <br/> <br/> **Script:** <br/> <script> <br/> var obj = { <br/> Id: 1, <br/> Subject: "Testing", <br/> StartTime: new Date(2018, 4, 5, 10, 00), <br/> EndTime: new Date(2018, 4, 5, 12, 00) <br/> }; <br/> var scheduleobj= \$("#schedule").ejSchedule(instance); <br/> scheduleobj.saveAppointment(obj); <br/> </script> | **Method:** *addEvent()* <br/><br/>  
 @Html.EJS().Schedule("schedule").EventSettings(e => e.DataSource(ViewBag.datasource)).Render() <br/> <br/> **Script:** <br/> <script> <br/> var scheduleobj= document.getElementById('schedule').ej2\_instances[0]; <br/> scheduleobj.addEvent({ <br/> Id: 1, <br/> Subject: 'New Event', <br/> StartTime: new Date(2018, 7, 31, 10, 30), <br/> EndTime: new Date(2018, 7, 31, 12, 0)}); <br/> </script> |  
 | To add resources dynamically | **Method:** *addResource* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields => fields.DataSource((IEnumerable)ViewBag.data))) <br/> <br/> **Script:** <br/> <script> <br/> var obj = { text: "Paul", id: 1, groupId: 3, color: "#cc99ff" }; <br/> var index = 0; <br/> var scheduleobj= \$("#schedule").ejSchedule(instance); <br/> scheduleobj.addResource(obj, "Owners", index); <br/> </script> | **Method:** *addResource()* <br/><br/> @Html.EJS().Schedule("schedule").EventSettings(e => e.DataSource(ViewBag.datasource)).Render() <br/> <br/> **Script:** <br/> <script> <br/> var obj = { text: "Paul", id: 3, groupId: 1, color: "#cc99ff" }; <br/> var index = 0; <br/> var scheduleobj=

```
document.getElementById('schedule').ej2_instances[0];
 scheduleobj.addResource(obj,
"Owners", index);
 </script> |
```

```
| databind | Not applicable | Method: dataBind()


```

```
@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.currentView = 'Month';
 scheduleobj.dataBind();
 </script> |
```

```
| Delete appointment manually | Method: deleteAppointment()


```

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.DataSource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var obj = {

 Id: 1,
 Subject: "Testing",
 StartTime: new Date(2018, 4, 5, 10, 00),
 EndTime:
new Date(2018, 4, 5, 12, 00)
 };
 var scheduleobj= $("#schedule").ejSchedule(instance);

 scheduleobj.deleteAppointment(obj);
 </script> | Method: deleteEvent()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var obj = {

Id: 1,
 Subject: "Testing",
 StartTime: new Date(2018, 4, 5, 10, 00),
 EndTime: new
Date(2018, 4, 5, 12, 00)
 };
 var scheduleobj=
document.getElementById('schedule').ej2_instances[0];
 scheduleobj.deleteEvent(obj) ;

 </script> |
```

```
| destroy scheduler | Method: destroy()


```

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.DataSource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 scheduleobj.destroy();
 </script> |
Method: destroy()

 @Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.destroy();
 </script> |
```

```
| Get cell details | Method: getSlotByElement()


```

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.DataSource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 var $td = $(".e-draggableworkarea
table tr td").first();
 var cellDetails = scheduleobj.getSlotByElement($td);
 </script> |
Method: getCellDetails()

 @Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var $td = $(".e-
content-table tbody tr td");
 var cellDetail = scheduleobj.getCellDetails($td);
 </script> |
```

```
| Get current view appointments | Method: getCurrentViewAppointments


```

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.DataSource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 var currApp =
scheduleobj.getCurrentViewAppointments();
 </script> | Method: getCurrentViewEvents()

 @Html.EJS().Schedule("schedule").EventSettings(e =>
```

```
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var currApp =
scheduleobj.getCurrentViewEvents();
 </script> |
```

```
| Get entire appointment collection | Method: getAppointments()

@Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.DataSource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 var AppDetails =
scheduleobj.getAppointments();
 </script> | Method: getEvents()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var AppDetails =
scheduleobj.getEvents();
 </script> |
```

```
| Get current view dates | Not applicable | Method: getCurrentViewDates()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var AppDetails =
scheduleobj.getCurrentViewDates();
 </script> |
```

```
| Get event details | Not applicable | Method: getEventDetails()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var AppDetails =
scheduleobj.getEventDetails(appElement);
 </script> |
```

```
| Get occurrences using event ID | Not applicable | Method: getOccurrenceByID()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var recCollection =
scheduleobj.getOccurrenceByID(1);
 </script> |
```

```
| Get occurrences in the provided date range | Not applicable | Method: getOccurrenceByRange()

 @Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var sDate = new
Date(2018, 1, 12);
 var eDate = new Date(2018, 1, 17);
 var recCollection =
scheduleobj.getOccurrenceByRange(sDate, eDate);
 </script> |
```

```
| Get resource details using index | Not applicable | Method: getResourceByIndex()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var resCollection =
scheduleobj.getResourceByIndex(2);
 </script> |
```

```
| Show spinner | Not applicable | Method: showSpinner()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
```

```
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.showSpinner();
 </script> |
```

| Hide spinner | Not applicable | **Method:** *hideSpinner()* <br><br>

```
@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.hideSpinner();
 </script> |
```

| Check whether the time slot is available | Not applicable | **Method:** *isSlotAvalible()* <br><br>

```
@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var sTime = new
Date(2018, 1, 14, 09, 30);
 var etime = new Date(2018, 1, 14, 10, 30);
 var resCollection
= scheduleobj.isSlotAvalible(sTime, eTime);
 </script> |
```

| Open the event window manually | Not applicable | **Method:** *openEditor()* <br><br>

```
@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];
 var $td = $(".e-
content-table tbody tr td");
 var cellDetail = scheduleobj.getCellDetails($td);

scheduleobj.openEditor(cellDetail);
 </script> |
```

| Refresh the scheduler | **Method** *refresh()* <br><br>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.Datasource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 scheduleobj.refresh();
 </script> |
Method: refresh()

 @Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.refresh();
 </script> |
```

| Refresh the events | **Method** *refreshAppointments()* <br><br>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.Datasource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 scheduleobj.refreshAppointments();

 </script> | Method: refreshEvents()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.refreshEvents();
 </script> |
```

| Refresh the scroller | **Method** *refreshScroller()* <br><br>

```
@(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.Datasource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 scheduleobj.refreshScroller();

</script> | Not Applicable |
```

| To remove resources dynamically | **Method** *removeResource()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>  
 fields.Datasource((IEnumerable)ViewBag.data))) <br> <br> **Script:** <br> <script> <br> var resID = 1;  
 <br> var scheduleobj= \$("#schedule").ejSchedule(instance); <br>  
 scheduleobj.removeResource(resID, "Owners"); <br> </script> | **Method:** *removeResource()*  
 <br/><br/> @Html.EJS().Schedule("schedule").EventSettings(e =>  
 e.DataSource(ViewBag.datasource)).Render() <br> <br> **Script:** <br> <script> <br> var obj = { text:  
 "Paul", id: 3, groupId: 1, color: "#cc99ff" }; <br> var index = 0; <br> var scheduleobj=  
 document.getElementById('schedule').ej2\_instances[0]; <br>  
 scheduleobj.removeResource(index, "Owners", ); <br> </script> |

| Export schedule as PDF | **Method** *exportSchedule()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>  
 fields.Datasource((IEnumerable)ViewBag.data))) <br> <br> **Script:** <br> <script> <br> var  
 scheduleobj= \$("#schedule").ejSchedule(instance); <br>  
 scheduleobj.exportSchedule("ActionName","null", null); <br> </script> | Not Applicable |

| Export scheduler events to ICS | **Method** *exportSchedule()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>  
 fields.Datasource((IEnumerable)ViewBag.data))) <br> <br> **Script:** <br> <script> <br> var  
 scheduleobj= \$("#schedule").ejSchedule(instance); <br>  
 scheduleobj.exportSchedule("ActionName","ExportToICS", null); <br> </script> | **Method:**  
*exportToCalendar()* <br/> @Html.EJS().Schedule("schedule").Render() <br/> <br> **Script:** <br>  
 <script><br>var scheduleobj= document.getElementById('schedule').ej2\_instances[0]; <br>  
 scheduleObj.exportToCalendar(); <br> </script> |

| Import scheduler events from ICS | Not Applicable | **Method:** *importCalendar()* <br/>  
 @Html.EJS().Schedule("schedule").Render() <br/> <br> **Script:** <br> <script><br>var scheduleobj=  
 document.getElementById('schedule').ej2\_instances[0]; <br> scheduleObj.importCalendar();  
 <br> </script> |

| Export scheduler appointments in Excel file | **Method** *exportToExcel()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>  
 fields.Datasource((IEnumerable)ViewBag.data))) <br> <br> **Script:** <br> <script> <br> var  
 scheduleobj= \$("#schedule").ejSchedule(instance); <br>  
 scheduleobj.exportToExcel("ActionName", null, true); <br> </script> | **Method:** *exportToExcel()*  
 <br/> @Html.EJS().Schedule("schedule").Render() <br/> <br> **Script:** <br> <script><br>var  
 scheduleobj= document.getElementById('schedule').ej2\_instances[0]; <br>  
 scheduleObj.exportToExcel(); <br> </script> |

| Print the scheduler | **Method** *print()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>  
 fields.Datasource((IEnumerable)ViewBag.data))) <br> <br> **Script:** <br> <script> <br> var  
 scheduleobj= \$("#schedule").ejSchedule(instance); <br> scheduleobj.print(); <br> </script> | Not  
 Applicable |

| Filter appointments | **Method** *filterAppointments()* <br/><br/>  
 @(Html.EJ().Schedule("schedule").AppointmentSettings(fields =>

```
fields.Datasource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 var filter = [{ field: "Subject", operator:
"contains", value: "with", predicate: "or" }];
 var filteredApp =
scheduleobj.filterAppointments(filter);
 </script> | Not Applicable |
```

```
| Search appointments | Method searchAppointments()

@Html.EJ().Schedule("schedule").AppointmentSettings(fields =>
fields.Datasource((IEnumerable)ViewBag.data)))

 Script:
 <script>
 var
scheduleobj= $("#schedule").ejSchedule(instance);
 var searchApp =
scheduleobj.searchAppointments("with");
 </script> | Not Applicable |
```

```
| To edit appointments manually | Not applicable | Method: saveEvent()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.saveEvent({
 Id: 1,
 Subject: 'Event edited',
 StartTime: new Date(2018,
7, 31, 10, 30),
 EndTime: new Date(2018, 7, 31, 12, 0)});
 </script> |
```

```
| Setting work hours | Not applicable | Method: setWorkHours()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.setWorkHours([new Date(2017, 9, 5)], '04:00', '08:00');
 </script> |
```

```
| Scrolling to specific time | Not applicable | Method: scrollTo()

@Html.EJS().Schedule("schedule").EventSettings(e =>
e.DataSource(ViewBag.datasource)).Render()

 Script:
 <script>
 var
scheduleobj= document.getElementById('schedule').ej2_instances[0];

scheduleobj.scrollTo('12:00');
 </script> |
```

## Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```
| Fires on the beginning of each scheduler action | Event: ActionBegin

@Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.ActionBegin("onActionBegin
"))

 Script:
 <script>
 function onActionBegin(args) {
 }
 </script> | Event:
ActionBegin


```

```
@Html.EJS().Schedule("schedule").ActionBegin("onActionBegin").Render()

 Script:

<script>
 function onActionBegin(args) {
 }
 </script> |
```

```
| Fires on the completion of each scheduler action | Event: ActionComplete

@Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.ActionComplete("onActionC
omplete"))

 Script:
 <script>
 function onActionComplete(args) {
 }

</script> | Event: ActionComplete


```

```
@Html.EJS().Schedule("schedule").ActionComplete("onActionComplete").Render()

Script:
 <script>
 function onActionComplete(args) {
 }
 </script> |
```

| Fires when the scheduler action gets failed | Not applicable | **Event:** *ActionFailure* <br/><br/>

@Html.EJS().Schedule("schedule").ActionFailure("onActionFailure").Render() <br><br> **Script:**  
<br> <script> <br> function onActionFailure(args) { <br> } <br> </script> |

| Fires on appointment hover | **Event:** *AppointmentHover* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.AppointmentHover("onAppointmentHover")))) <br><br> **Script:** <br> <script> <br> function onAppointmentHover(args) { <br> } <br> </script> | Not applicable |

| Fires before an appointment gets created | **Event:** *BeforeAppointmentCreate* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.BeforeAppointmentCreate("onBeforeAppointmentCreate")))) <br> <br> **Script:** <br><script> <br> function onBeforeAppointmentCreate(args) { <br> } <br> </script> | **Event:** *actionBegin* <br/><br/>  
@Html.EJS().Schedule("schedule").ActionBegin(onActionBegin).Render() <br> <br> **Script:**  
<br><script> <br> function onActionBegin(args) { <br> if(args.requestType == 'eventCreate')<br> } <br> </script> |

| Fires before an appointment gets edited | **Event:** *BeforeAppointmentChange* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.BeforeAppointmentChange("onBeforeAppointmentChange")))) <br> <br> **Script:** <br><script> <br> function onBeforeAppointmentChange(args) { <br> } <br> </script> | **Event:** *actionBegin* <br/><br/>  
@Html.EJS().Schedule("schedule").ActionBegin(onActionBegin).Render() <br> <br> **Script:**  
<br><script> <br> function onActionBegin(args) { <br> if(args.requestType == 'eventChange')<br> } <br> </script> |

| Fires before an appointment gets deleted | **Event:** *BeforeAppointmentRemove* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.BeforeAppointmentRemove("onBeforeAppointmentRemove")))) <br> <br> **Script:** <br><script> <br> function onBeforeAppointmentRemove(args) { <br> } <br> </script> | **Event:** *actionBegin* <br/><br/>  
@Html.EJS().Schedule("schedule").ActionBegin(onActionBegin).Render() <br> <br> **Script:**  
<br><script> <br> function onActionBegin(args) { <br> if(args.requestType == 'eventRemove')<br> } <br> </script> |

| Fires before the context menu opens on scheduler | **Event:** *BeforeContextMenuOpen* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.BeforeContextMenuOpen("onBeforeContextMenuOpen")))) <br> <br> **Script:** <br><script> <br> function onBeforeContextMenuOpen(args) { <br> } <br> </script> | Not applicable |

| Fires on cell click | **Event:** *CellClick* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.CellClick("onCellClick")))) <br> <br> **Script:** <br><script> <br> function onCellClick(args) { <br> } <br> </script> | **Event:** *CellClick* <br/><br/>  
@Html.EJS().Schedule("schedule").CellClick("onCellClick").Render() <br> <br> **Script:**  
<br><script> <br> function onCellClick(args) { <br> } <br> </script> |

| Fires on cell double click | **Event:** *CellDoubleClick* <br/><br/>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.CellDoubleClick("onCellDoubleClick")))) <br> <br> **Script:** <br><script> <br> function onCellDoubleClick(args) { <br> } <br> </script> | **Event:** *CellDoubleClick* <br/><br/>

@Html.EJS().Schedule("schedule").CellDoubleClick("onCellDoubleClick").Render() <br> <br>

**Script:** <br><script> <br> function onCellDoubleClick(args) { <br> } <br> </script> |

| Fires on cell hover | **Event:** *CellHover*

<br><br>@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.CellHover("onCellHover")))) <br> <br> **Script:** <br><script> <br> function onCellHover(args) { <br> } <br> </script> |

Not applicable |

| Fires once the scheduler is created | **Event:** *Create* <br><br>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.Create("onCreate")))

<br><br> **Script:** <br> <script> <br> function onCreate(args) { <br> } <br> </script> | **Event:** *Created*

<br><br> @Html.EJS().Schedule("schedule").Created("onCreated").Render() <br><br> **Script:**

<br> <script> <br> function onCreated(args) { <br> } <br> </script> |

| Fires on data binding action | Not applicable | **Event:** *DataBinding* <br><br>

@Html.EJS().Schedule("schedule").DataBinding("onDataBinding").Render() <br> <br> **Script:**

<br><script> <br> function onDataBinding(args) { <br> } <br> </script> |

| Fires after the data is bound to the control | Not applicable | **Event:** *DataBound* <br><br>

@Html.EJS().Schedule("schedule").DataBound("onDataBound").Render() <br> <br> **Script:**

<br><script> <br> function onDataBound(args) { <br> } <br> </script> |

| Fires once the scheduler is destroyed | **Event:** *destroy* <br><br>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.Destroy("onDestroy")))) <br>

<br> **Script:** <br><script> <br> function onDestroy(args) { <br> } <br> </script> | **Event:** *Destroyed*

<br><br> @Html.EJS().Schedule("schedule").Destroyed("onDestroyed").Render() <br> <br>

**Script:** <br> <script> <br> function onDestroyed(args) { <br> } <br> </script> |

| Fires on event click | **Event:** *AppointmentClick* <br><br>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.onAppointmentClick("AppointmentClick")))) <br><br> **Script:** <br> <script> <br> function onAppointmentClick(args) { <br> }

<br> </script> | **Event:** *EventClick* <br><br>

@Html.EJS().Schedule("schedule").EventClick("onEventClick").Render() <br> <br> **Script:**

<br><script> <br> function onEventClick { <br> } <br> </script> |

| Fires on event double click | **Event:** *AppointmentDoubleClick* <br><br>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.AppointmentDoubleClick("onAppointmentDoubleClick")))) <br> <br> **Script:** <br><script> <br> function

onAppointmentDoubleClick(args) { <br> } <br> </script> | Not applicable |

| Fires for keyboard actions | **Event:** *KeyDown* <br><br>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.KeyDown("onKeyDown")))

<br> <br> **Script:** <br> <script> <br> function onKeyDown(args) { <br> } <br> </script> | Not

applicable |

| Fires on context menu item click | **Event:** *MenuItemClick* <br><br>

@(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.MenuItemClick("onMenuItemClick")))) <br> <br> **Script:** <br><script> <br> function onMenuItemClick(args) { <br> } <br>

</script> | Not applicable |



| Fires on navigation | **Event:** *Navigation* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.Navigation("onNavigation")))  
 <br><br> **Script:** <br> <script> <br> function onNavigation(args) { <br> } <br> </script> | **Event:**  
*Navigating* <br/><br/> @Html.EJS().Schedule("schedule").Navigating("onNavigating").Render()  
 <br><br> **Script:** <br> <script> <br> function onNavigating(args) { <br> } <br> </script> |

| Fires on popup open | **Event:** *AppointmentWindowOpen* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.AppointmentWindowOpen("onAppointmentWindowOpen"))) <br><br> **Script:** <br> <script> <br> function  
 onAppointmentWindowOpen(args) { <br> } <br> </script> | **Event:** *PopupOpen* <br/><br/>  
 @Html.EJS().Schedule("schedule").PopupOpen("onPopupOpen").Render() <br> <br> **Script:**  
 <br><script> <br> function onPopupOpen(args) { <br> } <br> </script> |

| Fires on dragging event | **Event:** *Drag* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.Drag("onDrag"))) <br> <br>  
**Script:** <br><script> <br> function onDrag(args) { <br> } <br> </script> | **Event:** *Drag* <br/><br/>  
 @Html.EJS().Schedule("schedule").Drag("onDrag").Render() <br> <br> **Script:** <br><script> <br>  
 function onDrag(args) { <br> } <br> </script> |

| Fires on drag start | **Event:** *DragStart* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.DragStart("onDragStart"))) <br><br>  
**Script:** <br> <script> <br> function onDragStart(args) { <br> } <br> </script> | **Event:**  
*DragStart* <br/><br/> @Html.EJS().Schedule("schedule").DragStart("onDragStart").Render() <br>  
 <br> **Script:** <br><script> <br> function onDragStart(args) { <br> } <br> </script> |

| Fires on drag stop | **Event:** *DragStop* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.DragStop("onDragStop"))) <br><br>  
**Script:** <br><script> <br> function onDragStop(args) { <br> } <br> </script> | **Event:**  
*DragStop* <br/><br/> @Html.EJS().Schedule("schedule").DragStop("onDragStop").Render() <br>  
 <br> **Script:** <br><script> <br> function onDragStop(args) { <br> } <br> </script> |

| Fires on overflow button click | **Event:** *OverflowButtonClick* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.OverflowButtonClick("onOverflowButtonClick"))) <br> <br> **Script:** <br><script> <br> function onOverflowButtonClick(args) {  
 <br> } <br> </script> | **Event:** *popupOpen* <br/><br/>  
 @Html.EJS().Schedule("schedule").PopupOpen("onPopupOpen").Render() <br> <br> **Script:**  
 <br><script> <br> function onPopupOpen(args) { <br> if(args.requestType ==  
 'eventContainer')<br> } <br> </script> |

| Fires on overflow button hover | **Event:** *OverflowButtonHover* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.OverflowButtonHover("onOverflowButtonHover"))) <br> <br> **Script:** <br><script> <br> function onOverflowButtonHover(args)  
 { <br> } <br> </script> | Not applicable |

| Fires when the reminder action takes place | **Event:** *Reminder* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.Reminder("onReminder"))) <br><br>  
**Script:** <br><script> <br> function onReminder(args) { <br> } <br> </script> | Not  
 applicable |

| Fires on resizing event | **Event:** *Resize* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.Resize("onResize")))) <br>  
 <br> **Script:** <br><script> <br> function onResize(args) { <br> } <br> </script> | **Event:** *Resize*  
 <br/><br/> @Html.EJS().Schedule("schedule").Resize("onResize").Render() <br> <br> **Script:**  
 <br><script> <br> function onResize(args) { <br> } <br> </script> |

| Fires on resize start | **Event:** *ResizeStart* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.ResizeStart("onResizeStart"))) <br>  
 <br> **Script:** <br><script> <br> function onResizeStart(args) { <br> } <br> </script> | **Event:**  
*ResizeStart* <br/><br/> @Html.EJS().Schedule("schedule").ResizeStart("onResizeStart").Render()  
 <br><br> **Script:** <br> <script> <br> function onResizeStart(args) { <br> } <br> </script> |

| Fires on resize stop | **Event:** *ResizeStop* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.ResizeStop("onResizeStop"))) <br>  
 <br> **Script:** <br><script> <br> function onResizeStop(args) { <br> } <br> </script> | **Event:**  
*ResizeStop* <br/><br/> @Html.EJS().Schedule("schedule").ResizeStop("onResizeStop").Render()  
 <br> <br> **Script:** <br><script> <br> function onResizeStop(args) { <br> } <br> </script> |

| Fires on rendering of every scheduler elements | **Event:** *QueryCellInfo* <br/><br/>  
 @(Html.EJ().Schedule("schedule").ScheduleClientSideEvents(e=>e.QueryCellInfo("onQueryCellInfo"))) <br>  
 <br> **Script:** <br><script> <br> function onQueryCellInfo(args) { <br> } <br> </script> |  
**Event:** *RenderCell* <br/><br/>  
 @Html.EJS().Schedule("schedule").RenderCell("onRenderCell").Render() <br> <br> **Script:**  
 <br><script> <br> function onRenderCell(args) { <br> } <br> </script> <br> |

| Fires before the event rendering on UI | Not applicable | **Event:** *EventRendered* <br/><br/>  
 @Html.EJS().Schedule("schedule").EventRendered("onEventRendered").Render() <br> <br>  
**Script:** <br><script> <br> function onEventRendered(args) { <br> } <br> </script> |

You can refer to our [ASP.NET MVC Scheduler](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Scheduler](#) example to know how to present and manipulate data.

## How To

### Perform CRUD Actions Dynamically

CRUD actions can be manually performed on appointments using `addEvent`, `saveEvent` and `deleteEvent` methods as shown below.

#### Normal event

##### CSHTML

```
@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("ADD").Render()
 @Html.EJS().Button("btn2").Content("EDIT").Render()
 @Html.EJS().Button("btn3").Content("DELETE").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px"))
```

```

 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var Data = [{
 Id: 1,
 Subject: 'Conference',
 StartTime: new Date(2018, 1, 12, 9, 0),
 EndTime: new Date(2018, 1, 12, 10, 0),
 IsAllDay: false
 }, {
 Id: 2,
 Subject: 'Meeting',
 StartTime: new Date(2018, 1, 15, 10, 0),
 EndTime: new Date(2018, 1, 15, 11, 30),
 IsAllDay: false
 }];
 scheduleObj.addEvent(Data);
 document.getElementById('btn1').setAttribute('disabled', 'true');
 };
 document.getElementById('btn2').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var Data = {
 Id: 3,
 Subject: 'Testing-edited',
 StartTime: new Date(2018, 1, 11, 10, 0),
 EndTime: new Date(2018, 1, 11, 11, 0),
 IsAllDay: false
 };
 scheduleObj.saveEvent(Data);
 document.getElementById('btn2').setAttribute('disabled', 'true');
 };
 document.getElementById('btn3').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.deleteEvent(4);
 document.getElementById('btn3').setAttribute('disabled', 'true');
 };
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{

```

```

List<AppointmentData> appData = new List<AppointmentData>();
appData.Add(new AppointmentData
{
 Id = 3,
 Subject = "Testing",
 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
});
appData.Add(new AppointmentData
{
 Id = 4,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 13, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 13, 10, 0, 0),
 IsAllDay = false,
});
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

### Recurrence event

#### CSHTML

```

@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("ADD").Render()
 @Html.EJS().Button("btn2").Content("EDIT").Render()
 @Html.EJS().Button("btn3").Content("DELETE").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var Data = [{
 Id: 1,
 Subject: 'Conference',
 StartTime: new Date(2018, 1, 15, 9, 0),

```

```

 EndTime: new Date(2018, 1, 15, 10, 0),
 IsAllDay: false,
 RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=2'
 }];
 scheduleObj.addEvent(Data);
 document.getElementById('btn1').setAttribute('disabled', 'true');
 };
 document.getElementById('btn2').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var Data = new
ej.data.DataManager(scheduleObj.getCurrentViewEvents()).executeLocal(new
ej.data.Query().where(new ej.data.Predicate('RecurrenceID', 'equal', 3)));
 Data[0].EndTime = new Date(2018, 1, 11, 12, 0);
 scheduleObj.saveEvent(Data[0], 'EditOccurrence');
 document.getElementById('btn2').setAttribute('disabled', 'true');
 };
 document.getElementById('btn3').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var Data = {
 Id: 4,
 Subject: 'Vacation',
 RecurrenceID: 4,
 StartTime: new Date(2018, 1, 12, 11, 0),
 EndTime: new Date(2018, 1, 12, 12, 0),
 IsAllDay: false,
 RecurrenceRule: 'FREQ=DAILY;INTERVAL=1;COUNT=3'
 };
 scheduleObj.deleteEvent(Data, 'DeleteSeries');
 document.getElementById('btn3').setAttribute('disabled', 'true');
 };
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 3,
 Subject = "Testing",

```

```

 StartTime = new DateTime(2018, 2, 11, 9, 0, 0),
 EndTime = new DateTime(2018, 2, 11, 10, 0, 0),
 IsAllDay = false,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=3",
 });
 appData.Add(new AppointmentData
 {
 Id = 4,
 Subject = "Vacation",
 StartTime = new DateTime(2018, 2, 12, 11, 0, 0),
 EndTime = new DateTime(2018, 2, 12, 12, 0, 0),
 IsAllDay = false,
 RecurrenceRule = "FREQ=DAILY;INTERVAL=1;COUNT=2"
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public string RecurrenceRule { get; set; }
}

```

### Set Default Value for Event Fields

Event window default fields name like Title, Location, etc.. can be customized and default value can be set to Subject field using **Default** property which will be added if an appointment is created with empty subject.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(e => e.Fields(f =>
 f.Subject(sub => sub.Title("Event
Name").Name("Subject").Default("Add Name"))
 .Location(loc => loc.Title("Event Location").Name("Location"))
 .Description(des => des.Title("Summary").Name("Description"))
 .StartTime(st => st.Title("From").Name("StartTime"))
 .EndTime(et => et.Title("To").Name("EndTime"))
)
 .DataSource(ViewBag.datasource)
)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)

```

### DATA.CS

```

public ActionResult Index()
{

```

```

 ViewBag.datasource = GetScheduleData();
 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
 2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
 DateTime(2018, 2, 15, 9, 30, 0), EndTime = new DateTime(2018, 2, 15, 11, 0,
 0) });
 return appData;
 }

 public class AppointmentData
 {
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 }

```

## Open Editor Window in different ways

### Open Editor Window Manually

Scheduler allows the user to manually open the event editor on specific time or on certain events using **openEditor** method. To open the editor on specific range of time, user need to pass the cell details as first argument and **Add** as second argument whereas to open it on event pass that event detail and **Save** as arguments. In the following code example, on clicking the respective button will open the respective editor window manually.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("Click to open Editor").Render()
 @Html.EJS().Button("btn2").Content("Click to open Event
Editor").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .Views(ViewBag.view)
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];

```

```

 var cellData = {
 startTime: new Date(2018, 1, 15, 10, 0),
 endTime: new Date(2018, 1, 15, 11, 0),
 };
 scheduleObj.openEditor(cellData, 'Add');
 };
 document.getElementById('btn2').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var eventData = {
 Id: 3,
 Subject: 'Meteor Showers in 2018',
 StartTime: new Date(2018, 1, 14, 13, 0),
 EndTime: new Date(2018, 1, 14, 14, 30)
 };
 scheduleObj.openEditor(eventData, 'Save');
 };
</script>

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 9, 30, 0), EndTime = new DateTime(2018, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```



*Open editor window on single click*

By default, Scheduler Editor window will open when double clicking the cells or appointments. You can also open the editor window with single click by using `openEditor` method in `EventClick` and `CellClick` events of scheduler and setting false to `ShowQuickInfo`. The following example shows how to open editor window on single click of cells and appointments.

**CSHTML**

```
@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .ShowQuickInfo(false)
 .CellClick("onCellClick")
 .EventClick("onEventClick")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.appointments })
 .SelectedDate(new DateTime(2021, 7, 15))
 .Render()
)
<script type="text/javascript">
 function onCellClick(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 scheduleObj.openEditor(args, 'Add');
 }
 function onEventClick(args) {
 var scheduleObj = document.querySelector('.e-
schedule').ej2_instances[0];
 if (!(args.event).RecurrenceRule) {
 scheduleObj.openEditor(args.event, 'Save');
 }
 else {
 scheduleObj.quickPopup.openRecurrenceAlert();
 }
 }
</script>
```

**DATA.CS**

```
public ActionResult Index()
{
 ViewBag.appointments = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Paris",
 StartTime = new DateTime(2021, 7, 15, 10, 0, 0),
 EndTime = new DateTime(2021, 7, 15, 12, 30, 0)
 });
 return appData;
}
```

```
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}
```

### Prevent the Date Navigation

We can prevent navigation while clicking on the date header by simply removing `e-navigate` class from header cells which can be achieved in the `RenderCell` event as shown in the following code example.

#### CSHTML

```
@using Syncfusion.EJ2.Schedule
@(Html.EJS().Schedule("schedule")
 .Height("550px")
 .RenderCell("onRenderCell")
 .EventSettings(e => e.DataSource(ViewBag.datasource))
 .SelectedDate(new DateTime(2018, 1, 28))
 .Render()
)
<script type="text/javascript">
 function onRenderCell(args) {
 if(args.elementType === "dateHeader" || args.elementType ===
"monthCells") {
 ej.base.removeClass(args.element.childNodes, "e-navigate");
 }
 }
</script>
```

#### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 { Id = 1, Subject = "Blue Moon Eclipse", StartTime = new DateTime(2018,
2, 13, 9, 30, 0), EndTime = new DateTime(2018, 2, 13, 11, 0, 0) });
 appData.Add(new AppointmentData
 { Id = 2, Subject = "Milky Way as Melting pot", StartTime = new
DateTime(2018, 2, 15, 9, 30, 0), EndTime = new DateTime(2018, 2, 15, 11, 0,
0) });
 return appData;
}

public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
}
```

```

 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
}

```

### Half-yearly view

The year view of our scheduler displays all the 365 days and their related appointments of a particular year. You can customize the year view by using the following properties.

- [FirstMonthOfYear](#)
- [MonthsCount](#)
- [MonthHeaderTemplate](#)

In the following code example, you can see how to render only the last six months of a year in the scheduler. To start with the month of June, `FirstMonthYear` is set to 6 and `MonthsCount` is set to 6 to render only 6 months.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .FirstMonthOfYear(6)
 .MonthsCount(6)
 .MonthHeaderTemplate("#monthHeaderTemplate")
 .ResourceHeaderTemplate("#resourceHeaderTemplate")
 .Group(group => group.Resources(ViewBag.Resources))
 .Resources(res => {

res.AllowMultiple(true).DataSource(ViewBag.Owners).Field("OwnerId").Title("Owner").Name("Owners").TextField("OwnerText").IdField("Id").ColorField("OwnerColor").Add();
 })
 .Views(view => {
 view.Option(View.Year).Add();
 view.Option(View.TimelineYear).DisplayName("Horizontal Timeline Year").IsSelected(true).Add();
 view.Option(View.TimelineYear).DisplayName("Vertical Timeline Year").Orientation(Orientation.Vertical).Add();
 })
 .EventSettings(e => e.DataSource(ViewBag.datasources))
 .SelectedDate(new DateTime(2021, 8, 1))
 .Render()
)
<script id="monthHeaderTemplate" type="text/x-template">
 <div>${getMonthHeaderText(data.date)}</div>
</script>
<script id="resourceHeaderTemplate" type="text/x-template">
 <div class='template-wrap'>
 <div class="resource-details">
 <div class="resource-name">${getResourceName(data)}</div>
 </div>
 </div>
</script>

```

```

<script type="text/javascript">
 window.getMonthHeaderText = function (date) {
 var instance = new ej.base.Internationalization();
 return date.toLocaleString('en-us', { month: 'long' }) + ' ' +
date.getFullYear()
 };
 window.getResourceName = function (value) {
 return value.resourceData[value.resource.textField];
 };
</script>
<style>
 .e-schedule .e-vertical-view .e-resource-cells {
 height: 62px;
 }
 .e-schedule .template-wrap {
 display: flex;
 text-align: left;
 }
 .e-schedule .template-wrap .resource-details {
 padding-left: 10px;
 }
 .e-schedule .template-wrap .resource-details .resource-name {
 font-size: 16px;
 font-weight: 500;
 margin-top: 5px;
 }
 .e-schedule.e-device .template-wrap .resource-details .resource-name {
 font-size: inherit;
 font-weight: inherit;
 }
 .e-schedule.e-device .e-resource-tree-popup .e-fullrow {
 height: 50px;
 }
</style>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetResourceData();
 List<OwnerResource> owners = new List<OwnerResource>();
 owners.Add(new OwnerResource { OwnerText = "Nancy", Id = 1, OwnerColor =
"#ffaa00" });
 owners.Add(new OwnerResource { OwnerText = "Steven", Id = 2, OwnerColor
= "#f8a398" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 3, OwnerColor
= "#7499e1" });
 owners.Add(new OwnerResource { OwnerText = "Smith", Id = 4, OwnerColor =
"#5978ee" });
 owners.Add(new OwnerResource { OwnerText = "Michael", Id = 5, OwnerColor
= "#7499e1" });
 ViewBag.Owners = owners;
 ViewBag.Resources = new string[] { "Owners" };
 return View();
}
public List<ResourceData> GetResourceData()

```

```

{
 List<ResourceData> resourceData = new List<ResourceData>();
 resourceData.Add(new ResourceData
 {
 Id = 1,
 Subject = "Requirement planning",
 StartTime = new DateTime(2021, 9, 3, 10, 0, 0),
 EndTime = new DateTime(2021, 9, 3, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 1
 });
 resourceData.Add(new ResourceData
 {
 Id = 2,
 Subject = "Quality Analysis",
 StartTime = new DateTime(2021, 9, 4, 10, 0, 0),
 EndTime = new DateTime(2021, 9, 4, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 2
 });
 resourceData.Add(new ResourceData
 {
 Id = 3,
 Subject = "Resource planning",
 StartTime = new DateTime(2021, 9, 5, 10, 0, 0),
 EndTime = new DateTime(2021, 9, 5, 12, 0, 0),
 IsAllDay = false,
 OwnerId = 3
 });
 return resourceData;
}

public class ResourceData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
 public int OwnerId { get; set; }
}

public class OwnerResource
{
 public string OwnerText { set; get; }
 public int Id { set; get; }
 public string OwnerColor { set; get; }
}

```

### Set Different Working Hours on Different Days

By default, the work hours of the Scheduler is highlighted based on the start and end values provided within the `WorkHours` property which remains same for all days. To highlight different work hours range for different days, `setWorkHours` method. You can pass date object/ multiple date objects collection as first argument and start and end time need to be added as work hours should be passed as second and third arguments respectively. In the following code example, on button click 11:00 AM to 08:00 PM of 15th and 17th February has been added in work hours.

**CSHTML**

```

@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("Change the work hours").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .WorkHours(wh =>
 wh.Highlight(true)
 .Start("09:00")
 .End("11:00")
)
 .Views(ViewBag.view)
 .SelectedDate(new DateTime(2018, 2, 15))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var dates = [new Date(2018, 1, 15), new Date(2018, 1, 17)];
 scheduleObj.setWorkHours(dates, '11:00', '20:00');
 };
</script>

```

**DATA.CS**

```

public ActionResult Index()
{
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
}

```

[Show quick info Template](#)

This demo showcases the quick popups for cells and appointments with the customized templates.

**CSHTML**

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Schedule
<div class="control-section">
 <div class="content-wrapper">

@Html.EJS().Schedule("schedule").Width("100%").Height("650px").QuickInfoTemp
lates(new ScheduleQuickInfoTemplates { Header = "#header-template", Content

```

```

= "#content-template", Footer = "#footer-template"
}).PopupOpen("OnPopupOpen").EventRendered("onEventRendered").EventSettings(e
=> e.DataSource(ViewBag.datasources).SelectedDate(new DateTime(2020, 1,
09)).Resources(res =>
{
 res.DataSource(ViewBag.Categories).Field("RoomId").Title("Room
Type").Name("MeetingRoom").TextField("Name").IdField("Id").ColorField("Color
").Add();
}).Render()
</div>
</div>
<style>
.quick-info-header {
 background-color: white;
 padding: 8px 18px;
}
.quick-info-header-content {
 justify-content: flex-end;
 display: flex;
 flex-direction: column;
 padding: 5px 10px 5px;
}
.quick-info-title {
 font-weight: 500;
 font-size: 16px;
 letter-spacing: 0.48px;
 height: 22px;
}
.duration-text {
 font-size: 11px;
 letter-spacing: 0.33px;
 height: 14px;
}
.content-area {
 padding: 10px;
 width: auto;
}
.event-content {
 height: 90px;
 display: flex;
 flex-direction: column;
 justify-content: center;
 padding: 0 15px;
}
.meeting-type-wrap,
.meeting-subject-wrap,
.notes-wrap {
 font-size: 11px;
 color: #666;
 letter-spacing: 0.33px;
 height: 24px;
 padding: 5px;
}
.event-content div label {
 display: inline-block;
 min-width: 45px;
 color: #666;

```

```

 }
 .event-content div span {
 font-size: 11px;
 color: #151515;
 letter-spacing: 0.33px;
 line-height: 14px;
 padding-left: 8px;
 }
 .cell-footer .e-btn {
 background-color: #ffffff;
 border-color: #878787;
 color: #878787;
 }
 .cell-footer {
 padding-top: 10px;
 }
 .e-quick-popup-wrapper .e-cell-popup .e-popup-content {
 padding: 0 14px;
 }
 .e-quick-popup-wrapper .e-event-popup .e-popup-footer {
 display: block;
 }
 .e-quick-popup-wrapper .e-popup-footer button:first-child {
 margin-right: 5px;
 }
}
</style>
<script id="header-template" type="text/x-template">
 <div class="quick-info-header">
 <div class="quick-info-header-content"
style='${getHeaderStyles(data)}'>
 <div class="quick-info-title">${if (elementType == "cell")}Add
Appointment${else}Appointment Details${if}</div>
 <div class="duration-text">${getHeaderDetails(data)}</div>
 </div>
 </div>
</script>
<script id="content-template" type="text/x-template">
 <div class="quick-info-content">
 ${if (elementType == "cell")}
 <div class="e-cell-content">
 <div class="content-area">
 <input id="title" placeholder="Title" />
 </div>
 <div class="content-area">
 <input id="eventType" placeholder="Choose Type" />
 </div>
 <div class="content-area">
 <input id="notes" placeholder="Notes" />
 </div>
 </div>
 ${else}
 <div class="event-content">
 <div class="meeting-type-wrap">
 <label>Subject</label>:
 ${Subject}
 </div>
 <div class="meeting-subject-wrap">

```



```

 <label>Type</label>:
 ${getEventType (data) }
 </div>
 <div class="notes-wrap">
 <label>Notes</label>:
 ${Description}
 </div>
</div>
${/if}
</div>
</script>
<script id="footer-template" type="text/x-template">
 <div class="quick-info-footer">
 ${if (elementType == "cell")}
 <div class="cell-footer">
 <button id="more-details">More Details</button>
 <button id="add">Add</button>
 </div>
 ${else}
 <div class="event-footer">
 <button id="delete">Delete</button>
 <button id="more-details">More Details</button>
 </div>
 ${/if}
 </div>
</script>
<script type="text/javascript">
 var intl = new ej.base.Internationalization();
 window.getResourceData = function (data) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var resources = scheduleObj.getResourceCollections().slice(-1)[0];
 var resourceData = resources.dataSource.filter(function (resource) {
 return resource.Id === data.RoomId;
 })[0];
 return resourceData;
 };
 window.getHeaderDetails = function (data) {
 return intl.formatDate(data.StartTime, { type: 'date', skeleton:
'full' }) + ' (' +
 intl.formatDate(data.StartTime, { skeleton: 'hm' }) + ' - ' +
intl.formatDate(data.EndTime, { skeleton: 'hm' }) + ')';
 };
 window.getHeaderStyles = function (data) {
 if (data.elementType === 'cell') {
 return 'align-items: center; color: #919191;';
 }
 else {
 var resourceData = window.getResourceData(data);
 return 'background: ' + resourceData.Color + '; color: #FFFFFF;';
 }
 };
 var buttonClickActions = function (e) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var quickPopup = scheduleObj.element.querySelector('.e-quick-popup-
wrapper');

```

```

 var getSlotData = function () {
 var cellDetails =
scheduleObj.getCellDetails(scheduleObj.getSelectedElements());
 var addObj = {};
 addObj.Id = scheduleObj.getEventMaxID();
 addObj.Subject = quickPopup.querySelector('#title').value;
 addObj.StartTime = new Date(+cellDetails.startTime);
 addObj.EndTime = new Date(+cellDetails.endTime);
 addObj.Description = quickPopup.querySelector('#notes').value;
 addObj.RoomId =
quickPopup.querySelector('#eventType').ej2_instances[0].value;
 return addObj;
 };
 if (e.target.id === 'add') {
 var addObj = getSlotData();
 scheduleObj.addEvent(addObj);
 }
 else if (e.target.id === 'delete') {
 var eventDetails = scheduleObj.activeEventData.event;
 var currentAction = void 0;
 if (eventDetails.RecurrenceRule) {
 currentAction = 'DeleteOccurrence';
 }
 scheduleObj.deleteEvent(eventDetails, currentAction);
 }
 else {
 var isCellPopup = quickPopup.classList.contains('e-cell-popup');
 var eventDetail = isCellPopup ? getSlotData() :
 scheduleObj.activeEventData.event;
 var currentActions = isCellPopup ? 'Add' : 'Save';
 if (eventDetail.RecurrenceRule) {
 currentActions = 'EditOccurrence';
 }
 scheduleObj.openEditor(eventDetail, currentActions, true);
 }
 scheduleObj.closeQuickInfoPopup();
};
window.getEventType = function (data) {
 var resourceData = window.getResourceData(data);
 return resourceData.Name;
};
function onEventRendered(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 var categoryColor = args.data.CategoryColor;
 if (!args.element || !categoryColor) {
 return;
 }
 if (scheduleObj.currentView === 'Agenda') {
 (args.element.firstChild).style.borderColor = categoryColor;
 } else {
 args.element.style.backgroundColor = categoryColor;
 }
}
function OnPopupOpen(args) {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];

```

```

 if (args.type === 'QuickInfo') {
 var titleObj = new ej.inputs.TextBox({ placeholder: 'Title' });
 titleObj.appendTo(args.element.querySelector('#title'));
 var typeObj = new ej.dropdowns.DropDownList({
 dataSource: scheduleObj.getResourceCollections().slice(-
1) [0].dataSource,
 placeholder: 'Choose Type',
 fields: { text: 'Name', value: 'Id' },
 index: 0
 });
 typeObj.appendTo(args.element.querySelector('#eventType'));
 var notesObj = new ej.inputs.TextBox({ placeholder: 'Notes' });
 notesObj.appendTo(args.element.querySelector('#notes'));
 var moreDetailsBtn = args.element.querySelector('#more-
details');
 if (moreDetailsBtn) {
 var moreObj = new ej.buttons.Button({
 content: 'More Details', cssClass: 'e-flat',
 isPrimary:
args.element.firstElementChild.classList.contains('e-event-popup')
 });
 moreObj.appendTo(moreDetailsBtn);
 moreDetailsBtn.onclick = function (e) {
buttonClickActions(e); };
 }
 var addBtn = args.element.querySelector('#add');
 if (addBtn) {
 new ej.buttons.Button({ content: 'Add', cssClass: 'e-flat',
isPrimary: true }, addBtn);
 addBtn.onclick = function (e) { buttonClickActions(e); };
 }
 var deleteBtn = args.element.querySelector('#delete');
 if (deleteBtn) {
 new ej.buttons.Button({ content: 'Delete', cssClass: 'e-
flat' }, deleteBtn);
 deleteBtn.onclick = function (e) { buttonClickActions(e); };
 }
 }
 }
}
</script>

```

## DATA.CS

```

public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 List<QuickInfoDataSourceModel> categories = new
List<QuickInfoDataSourceModel>();
 categories.Add(new QuickInfoDataSourceModel { Name = "Jammy", Id
= 1, Capacity = 20, Color = "#ea7a57", Type = "Conference" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Tweety",
Id = 2, Capacity = 7, Color = "#7fa900", Type = "Cabin" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Nestle",
Id = 3, Capacity = 5, Color = "#5978ee", Type = "Cabin" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Phoenix",
Id = 4, Capacity = 15, Color = "#fec200", Type = "Conference" });
}

```

```

 categories.Add(new QuickInfoDataSourceModel { Name = "Mission",
Id = 5, Capacity = 25, Color = "#df5286", Type = "Conference" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Hangout",
Id = 6, Capacity = 10, Color = "#00bdae", Type = "Cabin" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Rick
Roll", Id = 7, Capacity = 20, Color = "#865fcf", Type = "Conference" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Rainbow",
Id = 8, Capacity = 8, Color = "#1aaa55", Type = "Cabin" });
 categories.Add(new QuickInfoDataSourceModel { Name = "Swarm", Id
= 9, Capacity = 30, Color = "#df5286", Type = "Conference" });
 categories.Add(new QuickInfoDataSourceModel { Name =
"Photogenic", Id = 10, Capacity = 25, Color = "#710193", Type = "Conference"
});

 ViewBag.Categories = categories;
 List<ScheduleView> viewOption = new List<ScheduleView>()
 {
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleView { Option = Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
 }
 public List<AppointmentData> GetScheduleData()
 {
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 RoomId= 10,
 Id= 1,
 Subject= 'Board Meeting',
 Description= 'Meeting to discuss business goal of 2020.',
 StartTime= '2020-01-05T04:00:00.000Z',
 EndTime= '2020-01-05T05:30:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 8,
 Id= 2,
 Subject= 'Training session on JSP',
 Description= 'Knowledge sharing on JSP topics.',
 StartTime= '2020-01-07T04:00:00.000Z',
 EndTime= '2020-01-07T05:30:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 3,
 Id= 3,
 Subject= 'Sprint Planning with Team members',
 Description= 'Planning tasks for sprint.',
 StartTime= '2020-01-09T04:00:00.000Z',
 EndTime= '2020-01-09T05:30:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 2,

```

```

 Id= 4,
 Subject= 'Meeting with Client',
 Description= 'Customer meeting to discuss features.',
 StartTime= '2020-01-11T03:30:00.000Z',
 EndTime= '2020-01-11T05:00:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 5,
 Id= 5,
 Subject= 'Support Meeting with Managers',
 Description= 'Meeting to discuss support plan.',
 StartTime= '2020-01-06T06:30:00.000Z',
 EndTime= '2020-01-06T08:00:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 1,
 Id= 6,
 Subject= 'Client Meeting',
 Description= 'Meeting to discuss client requirements.',
 StartTime= '2020-01-08T06:00:00.000Z',
 EndTime= '2020-01-08T07:30:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 10,
 Id= 1,
 Subject= 'Board Meeting',
 Description= 'Meeting to discuss business goal of 2020.',
 StartTime= '2020-01-05T04:00:00.000Z',
 EndTime= '2020-01-05T05:30:00.000Z'
 });
 appData.Add(new AppointmentData
 {
 RoomId= 7,
 Id= 7,
 Subject= 'Appraisal Meeting',
 Description= 'Meeting to discuss employee appraisals.',
 StartTime= '2020-01-10T05:30:00.000Z',
 EndTime= '2020-01-10T07:00:00.000Z'
 });
 return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public int RoomId { get; set; }
 public string StartTime { get; set; }
 public string EndTime { get; set; }
 public string Description { get; set; }
}
public class QuickInfoDataSourceModel
{
 public int Id { get; set; }
 public string Name { get; set; }
}

```

```

public int Capacity { get; set; }
public string Color { get; set; }
public string Type { get; set; }
}

```

### Enable scroll option on all-day section

When you have larger number of appointments in all-day row, it is difficult to view all the appointments properly. In that case you can enable scroller option for all-day row by setting true to **EnableAllDayScroll** whereas its default value is false. When setting this property to true, individual scroller for all-day row is enabled when it reaches its maximum height on expanding.

**Note:** This property is not applicable for Scheduler with Height **auto**.

### CSHTML

```

@using Syncfusion.EJ2.Schedule
@ (Html.EJS().Schedule("schedule")
 .Height("550px")
 .EventSettings(e => e.Fields(f => f.Subject(sub => sub.Name("Subject")))
 .Id("Id")
 .IsAllDay(allday => allday.Name("IsAllDay"))
 .StartTime(st => st.Name("StartTime"))
 .EndTime(et => et.Name("EndTime"))
)
 .DataSource(ViewBag.appointments)
)
.SelectedDate(new DateTime(2021, 3, 28))
.EnableAllDayScroll(true)
.Render()
)

```

### DATA.CS

```

public ActionResult Index()
{
 ViewBag.appointments = generateObject();
 return View();
}
public List<AppointmentData> generateObject()
{
 List<AppointmentData> appData = new List<AppointmentData>(25);
 for (int a = 0; a <= 25; a++)
 {
 appData.Add(new AppointmentData
 {
 Id = a + 1,
 Subject = 'Testing',
 StartTime = new Date(2021, 3, 28),
 EndTime = new Date(2021, 3, 29),
 IsAllDay = true
 });
 }
 return appData;
}
public class AppointmentData

```

```
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}
```

### Refresh Layout

In Scheduler, we can be able to refresh the layout manually without re-render the DOM element by using the `refreshLayout` public method. The following example code explains to know how to use the `refreshLayout` method.

### CSHTML

```
@using Syncfusion.EJ2.Schedule
<div>
 @Html.EJS().Button("btn1").Content("Refresh Layout").Render()
</div>
<div>
 @(Html.EJS().Schedule("schedule")
 .Width("100%")
 .Height("550px")
 .EventSettings(new ScheduleEventSettings { DataSource =
ViewBag.datasource })
 .SelectedDate(new DateTime(2021, 10, 15))
 .Render()
)
</div>
<script type="text/javascript">
 document.getElementById('btn1').onclick = function () {
 var scheduleObj =
document.getElementById('schedule').ej2_instances[0];
 scheduleObj.refreshLayout();
 };
</script>
```

### DATA.CS

```
public ActionResult Index()
{
 ViewBag.datasource = GetScheduleData();
 return View();
}
public List<AppointmentData> GetScheduleData()
{
 List<AppointmentData> appData = new List<AppointmentData>();
 appData.Add(new AppointmentData
 {
 Id = 1,
 Subject = "Conference",
 StartTime = new Date(2021, 10, 16, 10, 0),
 EndTime = new Date(2021, 10, 16, 12, 0),
 IsAllDay = false
 });
}
```

```

appData.Add(new AppointmentData
{
 Id = 2,
 Subject = "Meeting",
 StartTime = new Date(2021, 10, 18, 10, 0),
 EndTime = new Date(2021, 10, 18, 12, 30),
 IsAllDay = false
});
return appData;
}
public class AppointmentData
{
 public int Id { get; set; }
 public string Subject { get; set; }
 public DateTime StartTime { get; set; }
 public DateTime EndTime { get; set; }
 public bool IsAllDay { get; set; }
}

```

### Set Different Time Duration on Event Editor

In event window, start/end time duration will be processed based on the **interval** value within the **timeScale** property. By default, **interval** value is 30, therefore in event window start/end time duration will be in 30 mins duration. You can set custom interval range to the start/end time in event window using **popupOpen** event as shown below.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Schedule
<div class="control-section">
 <div class="content-wrapper">

@Html.EJS().Schedule("schedule").Width("100%").Height("550px").PopupOpen("on
PopupOpen").EventSettings(e =>
e.DataSource(ViewBag.datasource).Views(ViewBag.view).SelectedDate(new
DateTime(2018, 2, 15)).Render()
 </div>
 </div>
 <script type="text/javascript">
 function onPopupOpen(args) {
 args.duration = 40;
 }
 </script>

```

### DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public partial class ScheduleController : Controller

```



```

{
 public ActionResult event-duration()
 {
 ViewBag.datasource = new ScheduleData().GetScheduleData();
 List<ScheduleViewsModel> viewOption = new
List<ScheduleViewsModel>()
 {
 new ScheduleViewsModel {Option =
Syncfusion.EJ2.Schedule.View.Day },
 new ScheduleViewsModel {Option =
Syncfusion.EJ2.Schedule.View.Week },
 new ScheduleViewsModel {Option =
Syncfusion.EJ2.Schedule.View.WorkWeek },
 new ScheduleViewsModel {Option =
Syncfusion.EJ2.Schedule.View.Month }
 };
 ViewBag.view = viewOption;
 return View();
 }
}

```

### Prioritize the Resource Color for Events

By default top level resource color will be applied for the events. If user wants to apply specific resource color to events irrespective of its parent resource color, it can be achieved by `resourceColorField` field within `eventSettings` property as shown below.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Schedule
@section ControlsSection{
 <div class="control-section">
 <div class="content-wrapper">

@Html.EJS().Schedule("schedule").Width("100%").Height("550px").CurrentView(V
iew.WorkWeek).SelectedDate(new DateTime(2018, 6, 5)).Group(group =>
group.ByGroupID(false).Resources(ViewBag.Resources)).Resources(res =>
{

res.DataSource(ViewBag.Projects).Field("ProjectId").Title("Project").Name("P
rojects").TextField("text").IdField("id").ColorField("color").Add();

res.DataSource(ViewBag.Categories).Field("CategoryId").Title("Category").Nam
e("Categories").TextField("text").IdField("id").ColorField("color").AllowMul
tiple(true).Add();
 }).EventSettings(e => e.DataSource(ViewBag.datasource)).Render()
 </div>
 </div>
}

```

### DATA.CS

```

using System;
using System.Collections.Generic;

```

```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using WebApplication1.Models;
namespace WebApplication1.Controllers
{
 public partial class ScheduleController : Controller
 {
 public ActionResult resource-color()
 {
 ViewBag.datasource = new ScheduleData().GetResourceTeamData();
 List<ProjectResource> projects = new List<ProjectResource>();
 projects.Add(new ProjectResource { text = "PROJECT 1", id = 1,
color = "#cb6bb2" });
 projects.Add(new ProjectResource { text = "PROJECT 2", id = 2,
color = "#56ca85" });
 ViewBag.Projects = projects;
 List<CategoryResource> categories = new
List<CategoryResource>();
 categories.Add(new CategoryResource { text = "Development", id =
1, color = "#1aaa55" });
 categories.Add(new CategoryResource { text = "Testing", id = 2,
color = "#7fa900" });
 ViewBag.Categories = categories;
 ViewBag.Resources = new string[] { "Projects", "Categories" };
 return View();
 }
 }
 public class ProjectResource
 {
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 }
 public class CategoryResource
 {
 public string text { set; get; }
 public int id { set; get; }
 public string color { set; get; }
 }
}

```

**Note:** The `resourceColorField` field value should be as same as the `name` field value given with in `resources` property.

## SideBar

### Getting Started with ASP.NET MVC Sidebar Control

This section briefly explains about how to include [ASP.NET MVC Sidebar](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

<namespaces>

<add namespace="Syncfusion.EJ2"/>

</namespaces>

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### **~/ \_LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC

controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

#### Add ASP.NET MVC Sidebar control

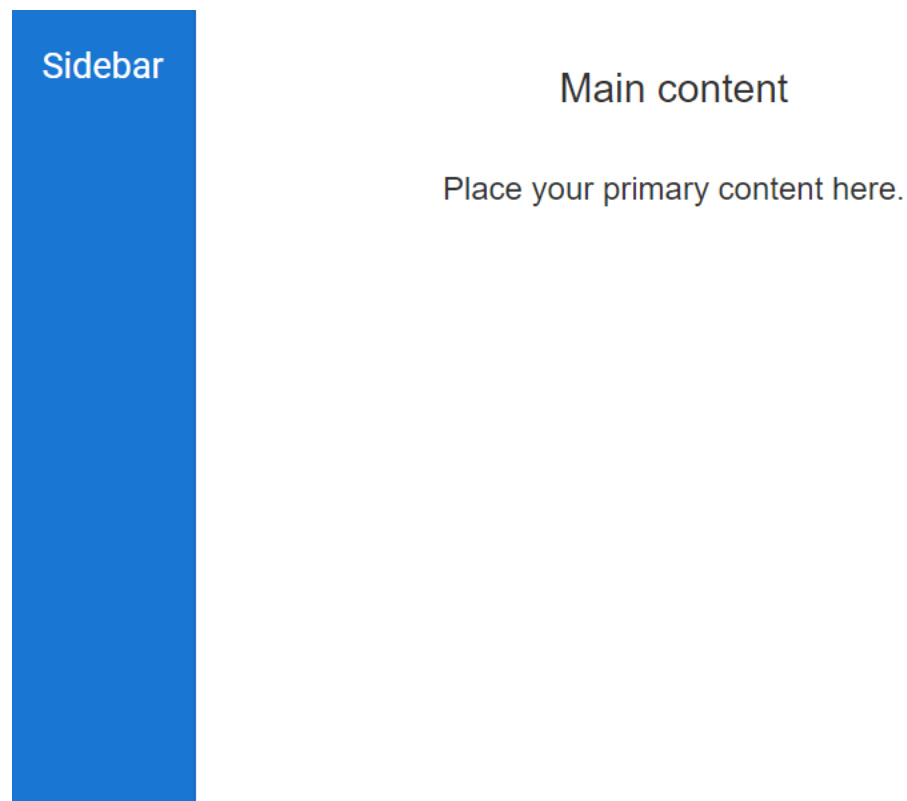
Now, add the Syncfusion ASP.NET MVC Sidebar control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").CloseOnDocumentClick(false).ContentTemplate(
@<div>
 <div class="title-header">
 <div style="display:inline-block"> Sidebar </div>
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- Main Content declaration -->
<div>
 <div class="title default">Main content</div>
 <div class="sub-title">
 Place your primary content here.
 </div>
</div>
<style>
/* sample level styles */
.center-align {
 text-align: center;
 padding: 20px;
}
.title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
}
.sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
}
body {
 padding-top: 0px;
 padding-bottom: 0px;
```

```
}
.title-header {
 text-align: center;
 font-size: 18px;
 padding: 15px;
}
.title.default {
 padding: 25px 15px 15px;
}
/* sidebar element styles */
#default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
}
</style>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Sidebar control will be rendered in the default web browser.



#### Enable backdrop

Enabling the [ShowBackdrop](#) in the Sidebar component will prevent the main content from user interactions. Here, the DOM elements will not get changed. It only closes the main content by covering with a black backdrop overlay and focuses the Sidebar in the screen. Sidebar can be rendered with specific width by setting [width](#) property.

**Note:** To achieve a proper **backdrop**, we suggest that you create a wrapper parent container for the div block in which you intend to enable the backdrop. Set the class name of this parent container as the **target** for the Sidebar. Alternatively, you can place an empty div container after the target container.

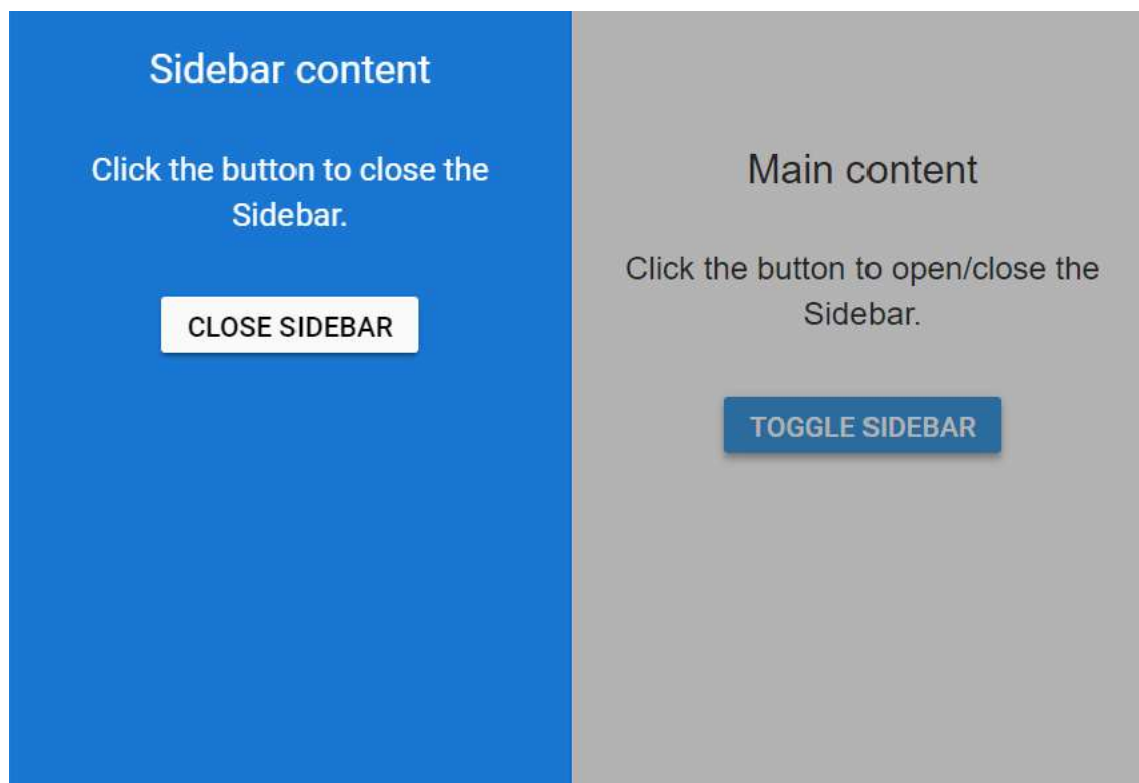
**CSHTML**

```

<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").ShowBackdrop(true).Type(Syncfusion.EJ2.Navigations.SidebarType.Push)
.Width("280px").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 Click the button to close the Sidebar.
 </div>
 <div class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("toggle").Content("Toggle Sidebar").CssClass("e-
info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 //create instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 // Toggle button to close and open the sidebar
 document.getElementById('toggle').onclick = function () {
 defaultSidebar.toggle();
 }
 // Close the sidebar
 document.getElementById('close').onclick = function () {
 defaultSidebar.hide();
 }
 });
</script>
<style>
 /* sample level styles */
 .center-align {
 text-align: center;
 padding: 20px;
 }
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;

```

```
padding: 10px;
}
/* Button styles */
#close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
background: #fafafa;
color: black
}
/* sidebar element styles */
#default-sidebar {
background-color: rgb(25, 118, 210);
color: #ffffff;
}
</style>
```



### Position

Positioning the Sidebar to the right or left of the main content can be achieved by using the [Position](#) property. If the position is not set, the Sidebar will expand from the left to the body element. [EnablePersistence](#) will persist the component's state between page reloads. [Change](#) event will be triggered when the state(expand/collapse) of the component is changed.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Type(Syncfusion.EJ2.Navigations.SidebarType.Push).Width("280px").E
nablePersistence(true).Target(".maincontent").ContentTemplate(@<div>
<div class="title"> Sidebar content</div>
<div class="sub-title">
```

```

 Click the button to close the Sidebar.
 </div>
 <div class="center-align">
 <!-- Button element declaration -->
 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- Main Content declaration -->
<div id="head">
 <!-- Button element declaration -->
 @Html.EJS().Button("toggle").Content("Open").IsToggle(true).CssClass("e-
info").IconCss("e-icons burg-icon").Render()
</div>
<div class="maincontent" style="height:335px;border:1px solid gray">
 <div>
 <div class="title">Main content</div>
 <div class="sub-title">
 <div class="column">
 <!-- RadioButton element declaration -->

 @Html.EJS().RadioButton("left").Label("Left").Name("state").Checked(true).Ch
ange("positionChange").Render()
 </div>
 <div class="column">
 <!-- RadioButton element declaration -->

 @Html.EJS().RadioButton("right").Label("Right").Name("state").Change("positi
onChange").Render()
 </div>
 </div>
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 //create instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 // Toggle button to close and open the sidebar
 document.getElementById('toggle').onclick = function () {
 var togglebtn =
document.getElementById("toggle").ej2_instances[0];
 if (document.getElementById('toggle').classList.contains('e-
active')) {
 togglebtn.content = 'Close';
 defaultSidebar.show();
 } else {
 togglebtn.content = 'Open';
 defaultSidebar.hide();
 }
 }
 // Close the sidebar
 document.getElementById('close').onclick = function () {
 defaultSidebar.hide();
 document.getElementById('toggle').classList.remove('e-active');

```



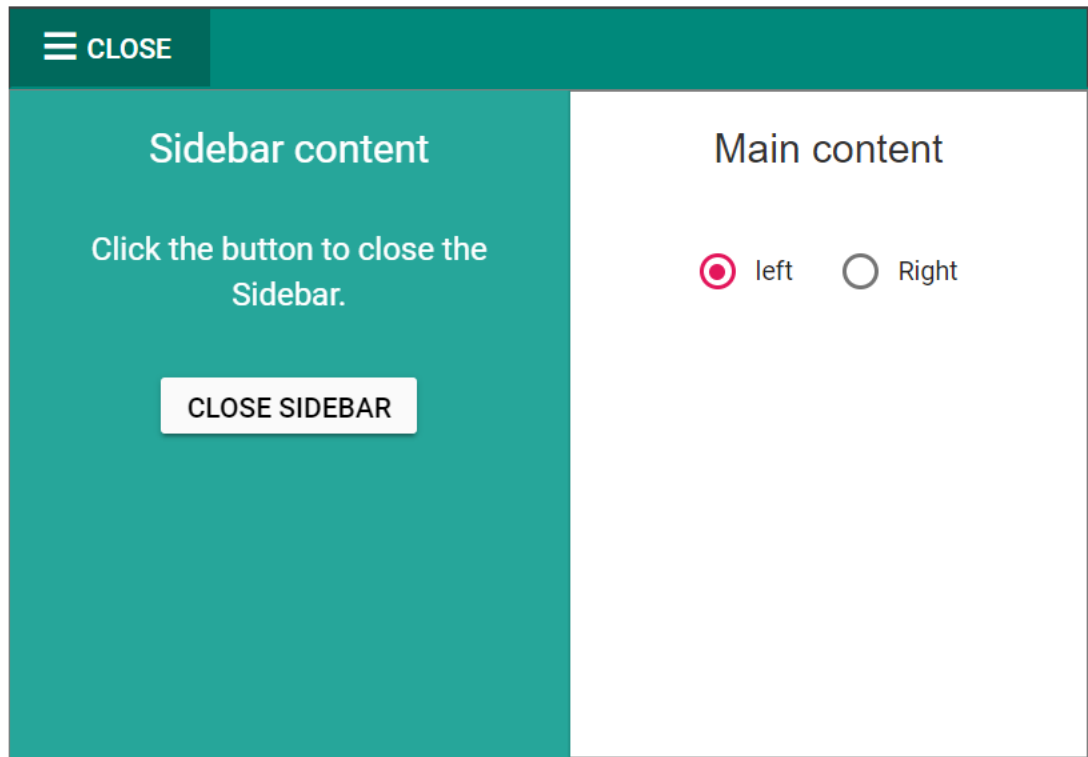
```

 var togglebtn =
document.getElementById("toggle").ej2_instances[0];
 togglebtn.content = 'Open'
 }
});
 // Change the position of sidebar
 function positionChange(args) {
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 defaultSidebar.position = (args.event.target.id == "left") ? "Left"
: "Right";
 }
</script>
<style>
 /* sample level styles */
 .center-align {
 text-align: center;
 padding: 20px;
 }
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 #head {
 border: 1px solid #424242;
 border-bottom-color: transparent;
 background: #00897B;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 .column {
 display: inline-block;
 padding: 10px;
 }
 /* Icons styles */
 .burg-icon:before {
 content: '\e10d';
 font-size: 16px;
 }
 /* Sidebar element styles */
 #default-sidebar {
 background-color: #26A69A;
 color: #ffffff;
 }
 /* Button styles */
 .close-btn:hover {
 color: #fafafa;
 }
 .radiobutton {
 display: inline-block;
 padding: 10px;
 }

```

```
#close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
}
#toggle { /* csslint allow: adjoining-classes*/
 background: #00695C;
 box-shadow: none;
 border-radius: 0;
 height: 39px;
 width: 100px;
}
/* custom generated icons styles */
@@font-face {
 font-family: 'e-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQukCgIAAAIkAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA/UD8wABAAAAAAAAAAAAAAAAADAABAAAAAQAAIb718
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAQAIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQpQAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAgAAAAAMAAAAUAAAAQA
AABQABABeAAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAGABWAKAAKABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAAAA+oD6gALAAATCQEXCQEHQCEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHESEVIREhFSE
VA9b8KgPW/CoD1vwq6I0B64wB640AAAEAAAAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBWYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHBoZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICakKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUEXIRERAOFXMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEHERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTEUUFhUWFxYWFxEQEBApDg4
NDQwMCwsKCgkICAgHBgYFBQQEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwmEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAgMDAwQE0AUEAWMDAgEBAgMDAwQ
FfaUEAWMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEBOAECawMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAWMDAgEBAgMDAwQFaAQEAwQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxyVExEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHBoZFxyVExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAEAOAQcuAsc+ATceAQEHBgcJgYPAQYWHwE
GFBCHDgEfAR4BPWEWHwEeATsBMjY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBLIYMhhSBhIIrgg
FAziFe4QJDADwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOEQqWMAAEAAAAAAwED6gAFAAAJAic
JAQEBaef+FhoBzf4za+v+Ff4VHwHMAc0AAAAAAQAAAAADAQpQAUAUAAEXCQEHAQL1Hf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAaAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUFHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFbQTEXITEREREBAPDw00DAw
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA
QERERERUUFbQTEXITEREREBAPDw00DAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwNDg8PERESEXQ
UFRYWFhgXGBkZGRoaGRkZGBcyFhYWFQRUEXIREQ8PDg0MCwoJCAcFBQMCAGMFBQcICQoLDA00Dw8
RERITFBQVFhYWGBcyGRkZGhoZGRkYFwgWfYVFBQTEHERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
```

```
RERAQEBAPDw8ODg4NDQIWcgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHM
TFBQUFRoZGRkYFwgWfHYVFBQTEHERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBu
WfHYFwgZGRkaGhkZGRgXGBYWFhUUFbMSEREpDw4NDAsKCQgHBQUdAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgwMDAsKCggIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICAcFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUdAgAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAAAAABAA0AAQABAAAAAAAAACAAcADgABAAAAAA
DAA0AFQABAAAAAAAAEAA0AIgABAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA
AAAAALABIAcWADAAEEECQAAAAIAhQADAAEEECQABABoAhwADAAEEECQACAA4AoQADAAEEECQADABoArWA
DAAEEECQAEABoAyQADAAEEECQAFABYA4wADAAEEECQAGABoA+QADAAEEECQAKAFgBEwADAAEEECQALACQ
BayB1LW1jb25zLW1ldHJvUmVndWxhcmtUtaWNvbnMtZWV0cm91LW1jb25zLW1ldHJvVmVyc2lvbiA
xLjB1LW1jb25zLW1ldHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAAC4AMABlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgBvAEYAbwBuAHQ
AIAbnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAAB
NAGUAdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBBQEgAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMtZWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbilzZWZ0CE1UX0NsZWfYDE1UX0plbmttYWlscwR
zdG9wAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
</style>
```



## Animate

Animation transitions can be set while expanding or collapsing the Sidebar using the [Animate](#) property. By default, [Animate](#) property is set to true. [EnableRtl](#) will display the sidebar in the right-to-left direction.

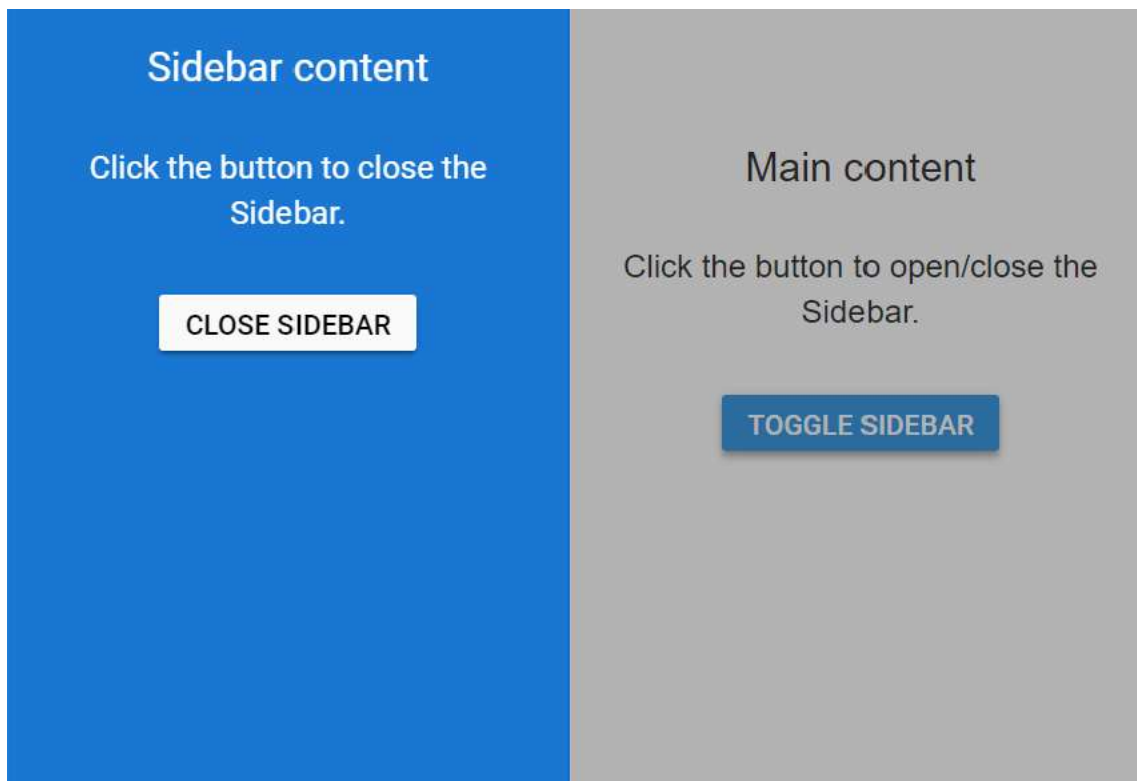
## CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Animate(false).EnableRtl(true).Type(Syncfusion.EJ2.Navigations.Sid
ebarType.Push).Width("280px").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 Click the button to close the Sidebar
 </div>
 <div class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("toggle").Content("Toggle Sidebar").CssClass("e-
info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 //create instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 // Toggle button to close and open the sidebar
 document.getElementById('toggle').onclick = function () {
 defaultSidebar.toggle();
 }
 // Close the sidebar
 document.getElementById('close').onclick = function () {
 defaultSidebar.hide();
 }
 });
</script>
<style>
 /* sample level styles */
 .center-align {
 text-align: center;
 padding: 20px;
 }
 .title {
 text-align: center;
```

```

 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 /* Button styles */
 #close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
 }
 /* sidebar element styles */
 #default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
</style>

```



#### Close on document click

SideBar can be closed on document click by setting [CloseOnDocumentClick](#) to true. If this property is not set, the SideBar will not close on document click since its default value is false. SideBar can be kept opened during rendering using [IsOpen](#) property.

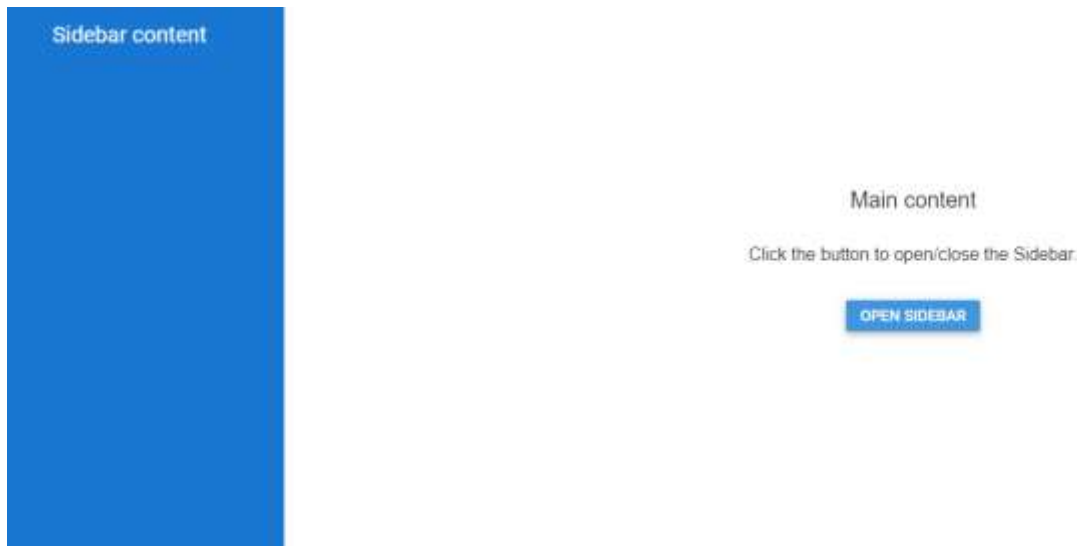
#### CSHTML

```
<!-- sidebar element declaration -->
```

```

@{Html.EJS().Sidebar("default-
sidebar").CloseOnDocumentClick(true).IsOpen(true).Type(Syncfusion.EJ2.Naviga
tions.SidebarType.Push).Width("280px").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("toggle").Content("Open Sidebar").CssClass("e-
info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 //create instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 // Toggle button to close and open the sidebar
 document.getElementById('toggle').onclick = function () {
 defaultSidebar.show();
 }
 });
</script>
<style>
 /* sample level styles */
 .center-align {
 text-align: center;
 padding: 20px;
 }
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 /* Button styles */
 #close, #close: hover, #close: active, #close: focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
 }
 /* sidebar element styles */
 #default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
</style>

```



### Enable gestures

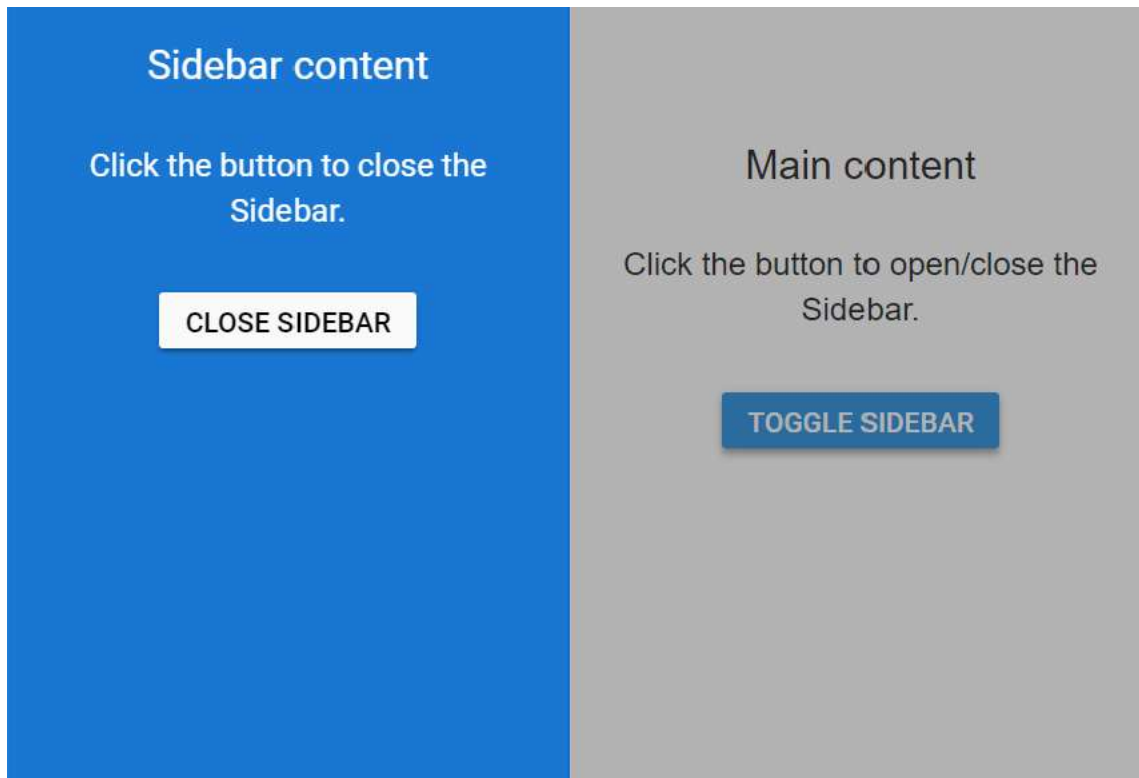
Expand or collapse the Sidebar while swiping in touch devices using [EnableGestures](#) property. By default, [EnableGestures](#) is set to true.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").EnableGestures(true).Type(Syncfusion.EJ2.Navigations.SidebarType.P
ush).Width("280px").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 Click the button to close the Sidebar
 </div>
 <div class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("toggle").Content("Toggle Sidebar").CssClass("e-
info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 //create instances for sidebar element
```

```
var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
// Toggle button to close and open the sidebar
document.getElementById('toggle').onclick = function () {
 defaultSidebar.toggle();
}
// Close the sidebar
document.getElementById('close').onclick = function () {
 defaultSidebar.hide();
}
});
</script>
<style>
/* sample level styles */
.center-align {
 text-align: center;
 padding: 20px;
}
.title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
}
.sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
}
/* Button styles */
#close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
}
/* sidebar element styles */
#default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
}
</style>
```





**Note:** [View Sample in GitHub.](#)

See also

- [Sidebar with Menu Component](#)
- [Sidebar Responsive Panel](#)
- [Usecase Sample](#)

## Context

By default, Sidebar initializes context to the body element. Using the [target](#) property, set context element to initialize Sidebar inside any HTML element apart from the body element.

**Note:** If required, `zIndex` can be set when sidebar act as overlay type.

In the following sample, click the toggle button to expand or collapse the sidebar and add button in sidebar element.

## CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Type(Syncfusion.EJ2.Navigations.SidebarType.Push).Width("280px").T
arget(".maincontent").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 Click the button to close the Sidebar.
 </div>
 <div class="center-align">
 <!-- Button element declaration -->
```

```

 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<div id="head">
 <!-- Button element declaration -->
 @Html.EJS().Button("toggle").Content("Open").IsToggle(true).CssClass("e-
info").IconCss("e-icons burg-icon").Render()
</div>
<!-- main content declaration -->
<div class="maincontent" style="height:335px;border:1px solid gray">
 <div>
 <div class="title">Main content</div>
 <div class="sub-title"> content goes here.</div>
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 // create a instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 // Show and Hide the sidebar
 document.getElementById('toggle').onclick = function () {
 var togglebtn =
document.getElementById("toggle").ej2_instances[0];
 if (document.getElementById('toggle').classList.contains('e-
active')) {
 togglebtn.content = 'Close';
 defaultSidebar.show();
 } else {
 togglebtn.content = 'Open';
 defaultSidebar.hide();
 }
 }
 // Close the sidebar
 document.getElementById('close').onclick = function () {
 var togglebtn =
document.getElementById("toggle").ej2_instances[0];
 defaultSidebar.hide();
 document.getElementById('toggle').classList.remove('e-active');
 togglebtn.content = 'Open'
 }
 });
</script>
<style>
 /* sample level styles */
 .center-align {
 text-align: center;
 padding: 20px;
 }
 #head {
 border: 1px solid #424242;
 border-bottom-color: transparent;
 background: #00897B;
 }
 body {

```

```

 margin: 0;
 }
 /* main content styles */
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 display: flex;
 justify-content: center;
 }
 }
 /* icon styles */
 .burg-icon:before {
 content: '\e10d';
 font-size: 16px;
 }
 /* Button styles */
 #toggle, #toggle:hover, #toggle:focus { /* csslint allow: adjoining-
classes*/
 background: #00695C;
 box-shadow: none;
 border-radius: 0;
 height: 39px;
 width: 100px;
 }
 #close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
 }
 /* Sidebar styles */
 #default-sidebar {
 background-color: #26A69A;
 color: #ffffff;
 }
 /* custom generated icons styles */
 @@font-face {
 font-family: 'e-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAkAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAA
wbG9jYQYkCgIAAAIkAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAACAAACAAAAAAAAAAAAAAEQIAAwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAAAAQpQAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAgAAAAAMAAAAUAMAAQA
AABQABABeAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAGABWAKAAkABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHESEVIREhFSE
VA9b8KgPW/CoDlVwq6I0B64wB640AAAEAAAAAA+oD4QCaaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs

```

```

CAA0aGhgMDAsLCwoKCgkJCQgHBWYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHB0zFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICakKCgs
LDAwNDQ4ODxAQEBEFxYWFhYVFRQUExIRERAOfxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTExUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBAQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDA wMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAGMDAwQE0AUEAWMDAgEBAGMDAwQ
FfaUEAWMDAgEBAGMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECawMDBAVoBAQDAwM
cAeUBAgIEAwQEaAUEAWMDAgEBAGMDAwQFAAQEAwQCAgElERMVFhcZGhwdHh4fICAhIiIhICAFHh4
dGxsZFxyVExEQDgsJCAUDAQEDBQCkCw4QERMVFhcZGxsHh4fICAhIiIhICAFHh4dHB0ZFxyVExE
QDgsKBwUDAQEDBQCkCw4AAAAIAAAAAA9MD6QALAE8AAAEQAQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBChDgEfAR4BPWEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGApS
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYMhSbHIIrgg
FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYuGYsCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4za+v+Ff4VHwHMAc0AAAAAAQAAAAADAQPqAAUAAAEXCQEAQL1Hf4zaC0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAAlDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ0ODg4PDw8QEBAQERERERUUFbQTExITEREREBAPDw0ODAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ0ODg4PDw8QEBA
QERERERUUFbQTExITEREREBAPDw0ODAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFVRQUExIREQ8PDg0MCwoJCACFBQMCAgMFBQCICQoLDA0ODw8
RERITFBQVfHwYGBcYGRkZGhoZGRkYFhgWfHwYVFBQTEHERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBAAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETeHMT
FBQUFRoZGRkYFhgWfHwYVFBQTEHERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBu
WFhYFhgZGRkaGhkZGRgXGBYWFhUUFBMSEREPdw4NDAsKCQgHBQUDAgIDBQUHCAkKCwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhBw8RDw8hLw8RPw6aAswNDgwMDAsKCgkIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICACFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICACFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUDAgAAABIA3gABAAAAAEEEEAAAAABAAAAAABAA0AAQABAAAAAAACAAcAdgABAAAAAA
DAA0AFQABAAAAAAEAA0AIgABAAAAAAFAAsALwABAAAAAAAGAA0AogABAAAAAAAKACwArwABAAA
AAAAALABIAcWADAAEECQAAAAIAhQADAAEECQABABOAhwADAAEECQACAA4AoQADAAEECQADABOArwA
DAAEECQAEABOAYQADAAEECQAFABYA4wADAAEECQAGABOa+QADAAEECQAKAFgBEwADAAEECQALACQ
BayB1LW1jb25zLW1ldHJvUmVndWxhcmtUtaWNvbnMtbnV0cm91LW1ldHJvVmVyc2lvbiA
xLjB1LW1ldHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIAB1AC0AaQBJAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwB1AC0AaQBJAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQBvAHMAaQBVAG4AIAAxAC4AMAB1AC0AaQBJAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgB1AHIAZQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBVAG4AIAAB
NAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBVAG4ALgBjAG8
AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAAAAAwBAGEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMtbnVudS0wMQR1c2VyB0JUX21uZm8pU2V0dGluZ19
BbmRyb2lkDWN0ZXZyb24tcmlnaHQMY2hldnJvbilzZWZ0CE1UX0NsZWfYDE1UX0plbmtdtYwlsCwR
zdG9wAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
</style>

```

## CONTEXT.CS

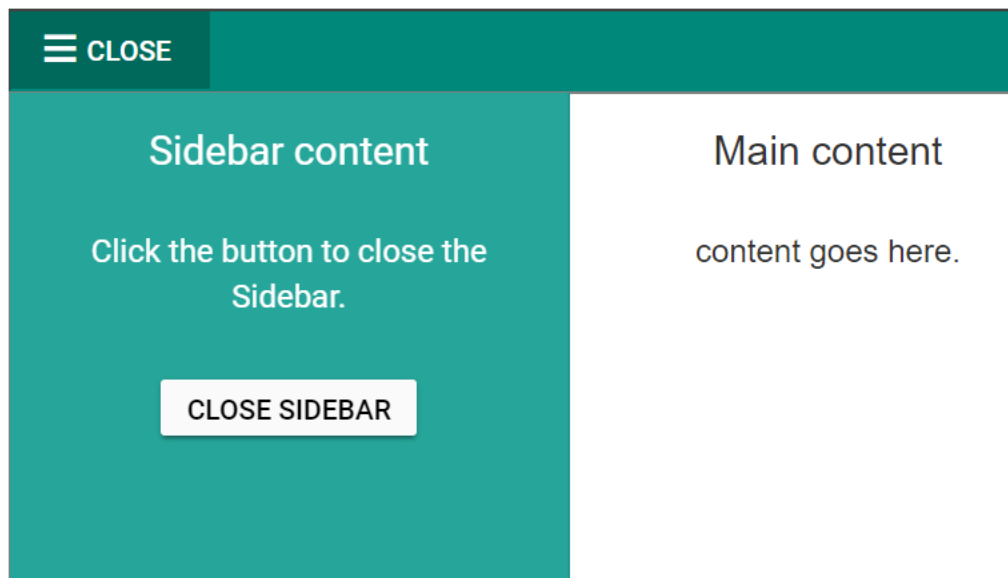
```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult context()
 {
 return View();
 }
 }
}

```

Output be like the below.



See Also

- [How to add layout sidebar](#)
- [How to add partial view sidebar](#)
- [Hide sidebar](#)

## Types

The Sidebar component's expand behaviour can be modified based on the purpose of use.

### Expanding types of Sidebar

The Sidebar can be set to initialize based on four different types that are consistent with the main component as explained below. When `dataBind` is invoked, this applies the pending property changes immediately to the component.

| Item | Description |
|------|-------------|
|      |             |

| ----- | -----  
----- |

| [Over](#) | Sidebar floats over the main content area.

| [Push](#) | Sidebar pushes the main content area to appear side-by-side, and shrinks the main content within the screen width.

| [Slide](#) | Sidebar translates the x and y positions of main content area based on the Sidebar width. The main content area will not be adjusted within the screen width. |

| [Auto](#) | Sidebar with [Over](#) type in mobile resolution, and [Push](#) type in other higher resolutions.

**Note:** [Auto](#) is the default expand mode.

In the following sample, Sidebar component's expand behaviour are demonstrated.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Type(Syncfusion.EJ2.Navigations.SidebarType.Push).Target(".maincon
tent").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 Click the button to close the Sidebar.
 </div>
 <div class="center-align">
 <!-- Close Button element declaration -->
 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<div id="head">
 <!-- Button element declaration -->
 @Html.EJS().Button("toggle").Content("Open").IsToggle(true).CssClass("e-
info").IconCss("e-icons burg-icon").Render()
</div>
<!-- Main Content declaration -->
<div class="maincontent" style="height:335px;border:1px solid gray">
 <div class="content">
 <div class="title">Main content</div>
 <div class="sub-title">
 <div class="radiobutton">
 <!-- RadioButton element declaration -->
 @Html.EJS().RadioButton("over").Label("Over").Name("state").Checked(true).Ch
ange("positionChange").Render()
 </div>
 <div class="radiobutton">
 <!-- RadioButton element declaration -->
 @Html.EJS().RadioButton("push").Label("Push").Name("state").Change("position
Change").Render()
 </div>
 </div>
 </div>
</div>
```

```

 <div class="radiobutton">
 <!-- RadioButton element declaration -->

@Html.EJS().RadioButton("slide").Label("Slide").Name("state").Change("positionChange").Render()
 </div>
 <div class="radiobutton">
 <!-- RadioButton element declaration -->

@Html.EJS().RadioButton("auto").Label("Auto").Name("state").Change("positionChange").Render()
 </div>
 </div>
</div>
<script type="text/javascript">
 // Toggle button to open and close the sidebar
 document.addEventListener('DOMContentLoaded', function () {

 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 var togglebtn = document.getElementById("toggle").ej2_instances[0];
 document.getElementById("toggle").addEventListener('click', function
() {
 if (document.getElementById('toggle').classList.contains('e-
active')) {
 togglebtn.content = 'Close';
 defaultSidebar.show();
 } else {
 togglebtn.content = 'Open';
 defaultSidebar.hide();
 }
 });
 // Close the sidebar
 document.getElementById("close").addEventListener('click', function
() {
 defaultSidebar.hide();
 document.getElementById('toggle').classList.remove('e-active');
 togglebtn.content = 'Open'
 });
 });
 // Change the position of sidebar
 function positionChange(args) {
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 if (args.event.target.id == "over") {
 defaultSidebar.type = "Over";
 defaultSidebar.dataBind();
 } else if (args.event.target.id == "push") {
 defaultSidebar.type = "Push";
 defaultSidebar.dataBind();
 }
 else if (args.event.target.id == "slide") {
 defaultSidebar.type = "Slide";
 defaultSidebar.dataBind();
 }
 else {

```

```

 defaultSidebar.type = "Auto";
 defaultSidebar.dataBind();
 }
}
</script>
<style>
/* sample level styles */
.center-align {
 text-align: center;
 padding: 20px;
}
.title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
}
.sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 display: flex;
 justify-content: center;
}
body {
 margin: 0;
}
/* icons styles */
.burg-icon:before {
 content: '\e10d';
 font-size: 16px;
}
/* Button element styles */
#head {
 border: 1px solid #424242;
 border-bottom-color: transparent;
 background: #00897B;
}
#toggle, #toggle:hover, #toggle:focus { /* csslint allow: adjoining-
classes*/
 background: #00695C;
 box-shadow: none;
 border-radius: 0;
 height: 39px;
 width: 100px;
}
#close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
}
.radiobutton {
 display: inline-block;
 padding: 10px;
}
/* Sidebar element styles */
#default-sidebar {
 background-color: #26A69A;

```



```

 color: #ffffff;
 }
 /* custom generated icons styles */
 @@font-face {
 font-family: 'e-icons';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcXfOQAAAKAAAAG4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAwbG9jYQYkCgIAAAIkAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbwAAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAAEAAAAAQIAAwAAAAAAgAAAAoACgAAAP8AAAAAAAAAAQpQAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAwGpZAJYAAAAABAAAAAAAAAAAAAPoAAA
D6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAAAAAgAAAAAUAAMAAQA
AABQABABeAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAQACAAYABAADAAGABwAKAAkABQALAAAAAAAAAB4AQABaQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAA+oD6gALAAATCQEXCQEHcQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUHESEVIREhFSE
VA9b8KgPW/CoDlVwq6I0B64wB640AAAEAAAAAA+oD4QCaaABMx8aHQEPDjEPah8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECaWUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHB0ZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDawMGRoD4gMEBwQFBQYGBwgICakKCgs
LDawNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOfxMLCggIBgYFBgQMDawNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEHERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTEUxUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBgQEBQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwydAg8GIS8
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM
DAwIBAQIDAwmEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAgMDAwQE0AUEAWMDAgEBAgMDAwQ
FfAUEAWMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRochR4eHyAgISI
iISAgHx4eHRSbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAgIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQE0AECawMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAWMDAgEBAgMDAwQFAwQEAWQCAGeLERMVfhcZGhwdHh4fICAhIiIhICAFHh4
dGxsZFxyVEXEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAFHh4dHB0ZFxyVEXE
QDgsKBwUDAQEDBQcKCw4AAAIAAAAAA9MD6QALAE8AAAE0AQcuASc+ATceAQEHBgcnJgYPAQYWHwE
GFBChDgEfAR4BPWEWHwEeATsBMjY/ATY3Fxy2Pwe2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYmhhSbHIIrgg
FAzIfe4QJDawJhBQeMgQGCK4IEgZSGDIYUGYsCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4za+v+Ff4VhWHMac0AAAAAAQAAAAADAQPqAAUAAAEEXCQEHAQLlHf4ZAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAALDw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFbQTExITEREREBAPDw00DAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA
QERERERUUFbQTExITEREREBAPDw00DAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYVFRQUExIREQ8PDg0MCwoJCACFBQMCAgMFBQCICQoLDA00Dw8
RERITFBQVfYhYwGBcYGRkZGhoZGRkYFhgWfYVFBQTEHERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEABAPDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehM
TFBQUFRoZGRkYFhgWfYVFBQTEHERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU
WfYyYFhgZGRKaGkZGRGyXGBYWFhUUFbMSEREPDw4NDAsKCQgHBQUdAgIDBQUHCAkKCwwNDg8PERE
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgWMDAsKCgIBwU
FAwIBAQIDBQUHCAgKCgsMDAwODf00DQ4MDAwLCgoICACFBQMCAQECAwUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICACFBQMCAQECAwUFBwgICgoLCwwNDQ4CzA4NDQwLCwo
KCAgHBQUdAgAAABIA3gABAAAAAAAAAAAAEAAAAABAAAAAABAA0AAQABAAAAAAACAAcAdgABAAAAAA
DAA0AFQABAAAAAAAEAA0AIgABAAAAAAFAAASALwABAAAAAAAGAA0AOgABAAAAAAAKACwArwABAAA
AAAAALABIAcWADAAEECQAAAAIAhQADAAEECQABAB0AhwADAAEECQACAA4AoQADAAEECQADAB0ArwA
DAAEECQAEAB0AyQADAAEECQAFABYA4wADAAEECQAGAB0A+QADAAEECQAKAFgBEwADAAEECQALACQ

```

```

BayB1LW1jb25zLW1ldHJvUmVndWxhcmtaWNvbnMtbWV0cm9lLW1jb25zLW1ldHJvVmVyc2lvbiA
xLjB1LW1jb25zLW1ldHJvRm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRYbyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwB1AC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgB1AHIAZQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBuaGMAZgB1AHMAaQBvAG4AIAB
NAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBuaGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAAwBAGEDAQQBBQEGAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQRlc2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHMY2hldnJvbilsZWZ0CE1UX0NsZWFiYDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
 }
</style>

```

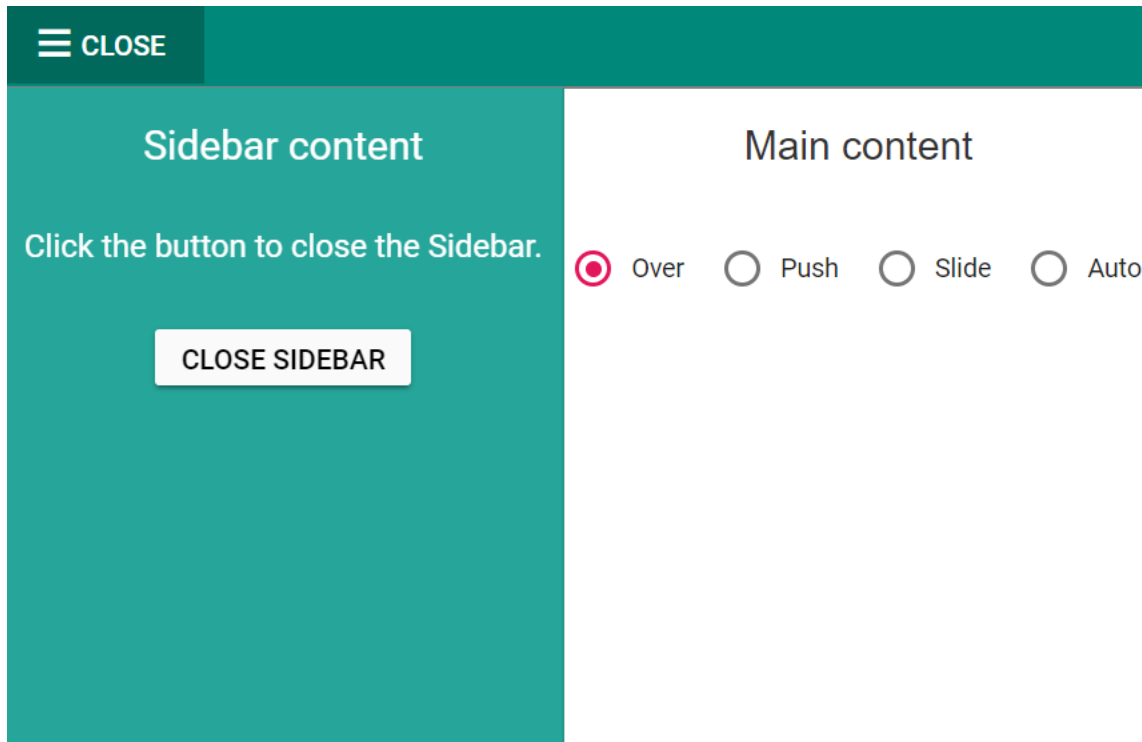
## TYPES.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult types()
 {
 return View();
 }
 }
}

```

Output be like the below.



See Also

- [How to add sidebar with custom animation](#)
- [How to add multiple sidebar](#)

### Auto-close

SideBar often behaves differently on a mobile versus a desktop display. It has an effective feature that offers to set it in opened or closed state corresponding to the specified resolution. This is achieved through [mediaQuery](#) property that allows you to set the SideBar in an expanded state or collapsed state only in user-defined resolution.

In the following sample, mediaQuery has been used for specific resolution to close and open sidebar.

### CSHTML

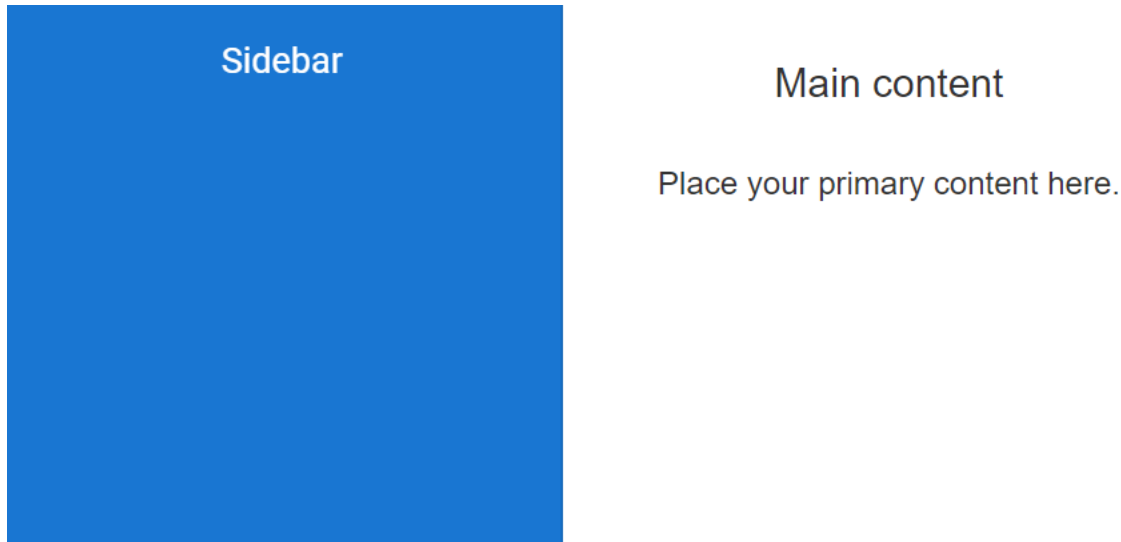
```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-sidebar").Width("280px").MediaQuery("(min-width: 600px)").ContentTemplate(@<div>
 <div class="title-header">
 <div style="display:inline-block"> Sidebar </div>
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title default">Main content</div>
 <div class="sub-title">
 Place your primary content here.
 </div>
</div>
```

```
</div>
<style>
 /* main content styles */
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 body {
 padding-top: 0px;
 padding-bottom: 0px;
 }
 /* sidebar element styles */
 #default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
 .title-header {
 text-align: center;
 font-size: 18px;
 padding: 15px;
 }
 .title.default {
 padding: 25px 15px 15px;
 }
</style>
```

### AUTOCLOSE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult autoclose()
 {
 return View();
 }
 }
}
```

Output be like the below.



- In this sample, the Sidebar will be in an expanded state only in resolution below 400px using [mediaQuery](#) for max-width.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-sidebar").Width("280px").MediaQuery("(max-width: 500px)").ContentTemplate(@<div>
 <div class="title-header">
 <div style="display:inline-block"> Sidebar </div>
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title default">Main content</div>
 <div class="sub-title">
 Place your primary content here.
 </div>
</div>
<style>
/* main content styles */
.title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
}
.sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
}
body {
 padding-top: 0px;
 padding-bottom: 0px;
}
```

```

/* sidebar element styles */
#default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
}
.title-header {
 text-align: center;
 font-size: 18px;
 padding: 15px;
}
.title.default {
 padding: 25px 15px 15px;
}
</style>

```

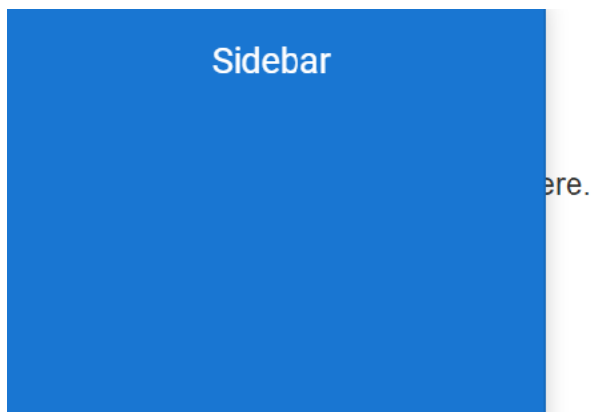
### AUTOCLOSE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult autoclose()
 {
 return View();
 }
 }
}

```

Output be like the below.



### Dock

Dock state of the Sidebar reserves some space on the page that always remains in a visible state when the Sidebar is collapsed. It is used to show the short term of a content like icons alone instead of lengthy text. To achieve this, set [EnableDock](#) as true along with required [DockSize](#).

In the following sample, the list item has icon with text representation. On dock state only the icon listed out to interact. It can be achieved by using [EnableDock](#) property.

### CSHTML

```

<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("dockSidebar").Width("220px").DockSize("72px").EnableDo
ck(true).ContentTemplate(@<div>
 <div class="dock">

 <li class="sidebar-item" id="toggle">

 Menu

 <li class="sidebar-item">

 Home

 <li class="sidebar-item">

 <span class="e-text"
title="profile">Profile

 <li class="sidebar-item">

 Info

 <li class="sidebar-item">

 <span class="e-text"
title="settings">Settings

 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div id="main-content container-fluid col-md-12 ">
 <div class="title">Main content</div>
 <div class="sub-title"> Click the expand icon to open and collapse icons
to close the Sidebar.</div>
</div>
<script>
 document.addEventListener('DOMContentLoaded', function () {
 // Toggle button to open and close the sidebar
 dockBar = document.getElementById("dockSidebar").ej2_instances[0];
 document.getElementById("toggle").addEventListener('click', function
() {
 dockBar.toggle();
 });
 });
</script>
<style>
 .title {
 text-align: center;
 font-size: 20px;
 }

```

```

 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 #wrapper .column {
 display: inline-block;
 padding: 10px;
 }
 .center {
 text-align: center;
 display: none;
 font-size: 13px;
 font-weight: 400;
 margin-top: 20px;
 }
 .sb-content-tab .center {
 display: block;
 }
 /* end of content area styles */
 /* Sidebar styles */
 .sb-content-tab #wrapper {
 display: none;
 }
 #dockSidebar.e-sidebar.e-right.e-close {
 visibility: visible;
 transform: translateX(0%);
 }
 #dockSidebar .e-icons::before {
 font-size: 25px;
 }
 /* dockbar icon Style */
 #dockSidebar .home::before {
 content: '\e102';
 }
 #dockSidebar .profile::before {
 content: '\e10c';
 }
 #dockSidebar .info::before {
 content: '\e11b';
 }
 #dockSidebar .settings::before {
 content: '\e10b';
 }
 #dockSidebar.e-sidebar .expand::before,
 #dockSidebar.e-sidebar.e-right.e-open .expand::before {
 content: '\e10f';
 }
 #dockSidebar.e-sidebar.e-open .expand::before,
 #dockSidebar.e-sidebar.e-right .expand::before {
 content: '\e10e';
 }
 /* end of dockbar icon Style */
 #dockSidebar.e-close .sidebar-item {
 padding: 5px 20px;
 }

```



```

}
#dockSidebar.e-dock.e-close span.e-text {
 display: none;
}
#dockSidebar.e-dock.e-open span.e-text {
 display: inline-block;
}
#dockSidebar li {
 list-style-type: none;
 cursor: pointer;
}
#dockSidebar ul {
 padding: 0px;
}
#dockSidebar.e-sidebar ul li:hover span {
 color: white
}
#dockSidebar.e-sidebar.e-open .e-text {
 overflow: hidden;
 text-overflow: ellipsis;
 line-height: 23px;
 font-size: 15px;
}
#dockSidebar.e-sidebar.e-open .e-icons {
 margin-right: 16px;
}
#dockSidebar.e-sidebar span.e-icons {
 color: #c0c2c5;
 line-height: 2
}
#dockSidebar.e-sidebar .e-open .e-icons {
 margin-right: 16px;
}
#dockSidebar.e-sidebar .e-open .e-text {
 overflow: hidden;
 text-overflow: ellipsis;
 line-height: 23px;
 font-size: 15px;
}
.sidebar-item {
 text-align: center;
 border-bottom: 1px #e5e5e5 solid;
}
#dockSidebar.e-sidebar.e-open .sidebar-item {
 text-align: left;
 padding-left: 15px;
 color: #c0c2c5;
}
#dockSidebar.e-sidebar {
 background: #2d323e;
 overflow: hidden;
}
@@font-face {
 font-family: 'e-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAAAAwAgTlMvMjciQ6oAAAEoAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAkAAAAG4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA

```

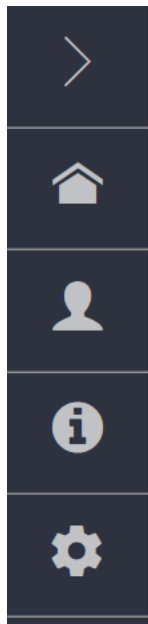
wbG9jYQukCgIAAAIkAAAAAGm1heHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw  
AAAZoAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8  
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAACAACAAAAAAAAAAAEAAAAAQIAAwAAAA  
AAgAAAAoACgAAP8AAAAAAAAAAQpQAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAwGpZAJYAAAAABAAAAAAAAABAAAAAPoAAA  
D6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAA+gAAAPoAAD6AAAAAAAgAAAAAMAAAAUAMAAQA  
AABQABABeAAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA  
AAAABAA4ADgAOABYAFgAYAAAAAQACAAYABAADAAGABWAKAAKABQALAAAAAAAAAB4AQABaAQYB5gJ  
kAnoCjgKwA8oEHAAAAIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D  
+lwFpAcP+s24BTf6qbgAAAAEAAAAA+oD6gALAAATCQEXCQEHQEnCQF4AYgBiGP+eAGIY/54/nh  
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABWALAAA3IRUheSEVIREhFSE  
VA9b8KgPW/CoDlVwq6I0B64wB640AAAEAAAAA+oD4QCaAAABMx8aHQEPDjEPAh8bIT8bNS8SPxs  
CAA0aGhgMDAsLCwoKCgkJCQgHBWYGBgUEBAMCAgECaWUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI  
dHh0cHB0ZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAwIBAQMEDg0NDAs  
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDawMGRoD4gMEBwQFBQYGBwgICakKCgs  
LDawNDQ4ODxAQEBEWFxYWFhYVFRQUEXIRERAOFxMLCgGIBgYFBgQMDawNDg4QDxERERIJCQkKCQk  
JFRQJCQoJCQgJEHERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTEUUFhUWFxYWFxEQEBApDg4  
NDQwMCwsKCgkICAgHBgYFBQQEBAQAAAAAawAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8  
GPQI/BjM1KwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8ffz8XLxcPBgI+BQQDAwMCAT8EBAM  
DAwIBAQIDawMEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAgMDawQE0AUEAWMDAgEBAgMDawQ  
FfaUEAWMDAgEBAgMDawQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRocHR4eHyAgISI  
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY  
BAgIDBAQE/rMBAQEDawQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEB0AECawMDBAVoBAQDAwM  
CAeUBAgIEAwQEaAUEAWMDAgEBAgMDawQFaAQEAwQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4  
dGxsZFxYVExEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHB0ZFxYVExE  
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAA9MD6QALAE8AAAEAOAQcuASc+ATceAQEHBgcJgYPAQYWHWE  
GFBCHDgEfAR4BPWEWHwEeATsBMjY/ATY3FxY2PwE2Ji8BNjQnNz4BLwEuAQ8Bji8BLgErASIGaps  
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh  
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYmhsBhIIRgg  
FAzIfe4QJDawJhBQeMgQGCK4IEgZSGDIYUGYsCK4IBQMyHxOECQwMAAEAAAAAwED6gAFAAAJAic  
JAQEBaef+Fh0Bzf4za+v+Ff4VHwHMAc0AAAAAAQAAAAADAQpQAUAUAAEXCQEHAQlHf4zAc0a/hY  
D6x7+M/40HwHrAAEAAAAA/MD8wALAAATCQEXCQE3CQENcQENAY7+cmQBjwGPZP5yAY5k/nH+cQO  
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEEAgQEBAAlDw4rAS8dPQE/DgUVDw4BPw4  
7AR8dBRUFHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFbQTExITEREREBAPDw00DAw  
LCwkJCAcGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA  
QERERERUUFbQTExITEREREBAPDw00DAwLCwkJCAcGBgQEAgL8fgIDBQUHCAkKCwNDg8PERESExQ  
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFQRUEXIREQ8PDg0MCwoJCAcFBQMCAgMFBQcICQoLDA00Dw8  
RERITFBQVfYhYWGBCYGRkZGhoZGRkYFxfGfYhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQcHBgY  
FBQMEAgIBAgIEBAYGBwgJCQsLDawODQ8PEBARERETeHMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE  
RERAQEBApDw8ODg4NDQIwCgkICQcHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDawODQ8PEBARERETeHMT  
FBQUFRoZGRkYFxfGfYhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsMDQ4PDxEREhMUFBU  
WFhYFxfGfYhYVFBQTEhERDw8ODQwLCgkIBwUFAwUUFbMSEREpDw4NDAsKCQgHBQUDAgIDBQUHCAkKCwNDg8PERE  
SExQUFRYWFhgXGBkZGQAAAQAAAAAD6gPqAEMAABMhHw8RDw8hLw8RPw6aAswNDgWMDAsKCgGIBwU  
FAwIBAQIDBQUHCAgKCgsMDawODf00DQ4MDawLCgoICAcFBQMCAQECAwUFBwgICgoLDawMDgPrAQI  
DBQUHCAgKCgsLDA0NDv00Dg0NDAsLCgoICAcFBQMCAQECAwUFBwgICgoLCwWNDQ4CzA4NDQwLCwo  
KCAgHBQUDAgAAABIA3gABAAAAAAAAAAEAAAABAAAAAAAAABAA0AAQABAAAAAAAAACAACADgABAAAAAA  
DAA0AFQABAAAAAAAEAA0AIgABAAAAAAFAAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwARwABAAA  
AAAAALABIAcWADAEEECQAAAAIAhQADAEEECQABAB0AhWADAEEECQACAA4AoQADAEEECQADAB0ArWA  
DAAEECQAEAB0AyQADAEEECQAFABYA4wADAEEECQAGAB0A+QADAEEECQAKAFgBEWADAEEECQALACQ  
BayB1LW1jb25zLW1ldHJvUmVndWxhcmUtaWNvbnMtbWV0cm91LW1ldHJvVmVyc2lvbiA  
xLjB1LW1ldHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV  
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAFIAZQBnAHU  
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbwBlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgB  
vAFYAZQByAHMAaQBvAG4AIAAxAAC4AMABlAC0AaQBjAG8AbgBzAC0AbQBlAHQAcgBvAEYAbwBuAHQ  
AIABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgBlAHMAaQBvAG4AIAIB  
NAGUAdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgBlAHMAaQBvAG4ALgBjAG8  
AbQAAAAACAAAAAAAOAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAwBAgEDAQQBQEGAQCBCAEJAQo

```
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMHbWVudS0wMQR1c2VyB0JUX2luZm8PU2V0dGluZ19BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hldnJvbilzZWZ0CE1UX0NsZWFiYDE1UX0p1bmttYWlscwRzdG9wAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* end of sidebar styles */
</style>
```

## DOCK.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult dock()
 {
 return View();
 }
 }
}
```

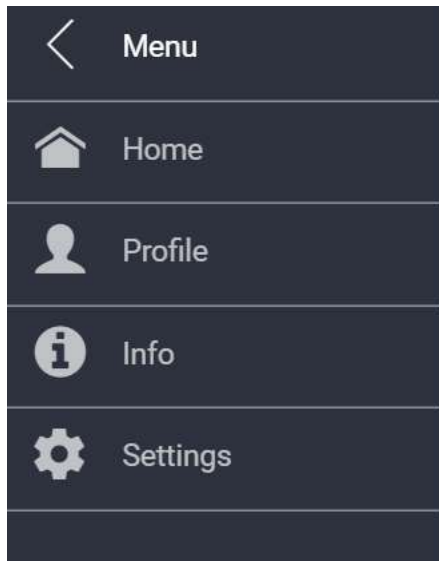
In Collapsed state,the output be like the below



Main content

Click the expand icon to open and collapse icons to close the Sidebar

In Expanded state, the output be like the below



## Main content

Click the expand icon to open and collapse icons to close the Sidebar

See Also

- [How to add sidebar navigation](#)

## Styles and Appearance

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user's preference.

### Customizing the sidebar

Use the below CSS to customize the sidebar root element.

```
`css
.e-sidebar {
background: #898b2b
}
`
```

### Customizing the sidebar based on the positions

Use the below CSS to customize the left positioned sidebar.

```
`css
.e-sidebar.e-left {
border-right: 2px solid red;
}
`
```

Use the below CSS to customize the right positioned sidebar.

```
`css
.e-sidebar.e-right {
```

```
border-left: 2px solid red;
}
```

,

#### Customizing the sidebar based on the active state

Use the below CSS to customize the open state of the left positioned sidebar.

```
`css
.e-sidebar.e-left.e-open {
transition: transform 2.5s ease;
}
```

,

Use the below CSS to customize the open state of the right positioned sidebar.

```
`css
.e-sidebar.e-right.e-open {
transition: transform 2.5s ease;
}
```

,

Use the below CSS to customize the closed state of the left positioned sidebar.

```
`css
.e-sidebar.e-left.e-transition.e-close {
transition: transform 2.5s ease, visibility 1200ms;
}
```

,

Use the below CSS to customize the closed state of the right positioned sidebar.

```
`css
.e-sidebar.e-right.e-transition.e-close {
transition: transform 2.5s ease, visibility 1200ms;
}
```

,

#### Customizing the sidebar with dock state

When you enable the Dock support, the "e-dock" class will be added to the root element. Based on that class, you can also customize all the above stated customization. Use the following CSS to customize the sidebar element with a dock state.

```
`css
.e-sidebar.e-dock {
```

```
background: #2d323e;
}
```

,

### Customizing the different types of sidebar

Use the below CSS to customize the auto type sidebar.

```
`css
```

```
.e-sidebar.e-left.e-auto {
background-color: pink;
}
```

,

Use the below CSS to customize the push type sidebar.

```
`css
```

```
.e-sidebar.e-left.e-push {
background-color: beige;
}
```

,

Use the below CSS to customize the over type sidebar.

```
`css
```

```
.e-sidebar.e-left.e-over {
background-color: aqua;
}
```

,

Use the below CSS to customize the slide type sidebar.

```
`css
```

```
.e-sidebar.e-left.e-slide {
background-color: green;
}
```

,

### Customizing the backdrop of the sidebar

Use the below CSS to customize the backdrop of the sidebar.

```
`css
```

```
.e-sidebar-overlay {
background-color: aqua;
}
```

### Customizing the sidebar in the RTL direction

When you enable the RTL (right to left direction) support, the "e-rtl" class will be added to the root element. Based on that class, you can also customize all the above stated customization. Use the following CSS to customize the sidebar element in the RTL (right to left direction) mode.

```
`css
.e-sidebar.e-left.e-rtl {
background-color: antiquewhite;
}
```

### Prevent the animation transition for the Sidebar component

Use the below CSS to prevent the animation transition for the Sidebar component.

```
`css
.e-sidebar-context .e-content-animation {
transition: none !important;
transform: none !important;
}
```

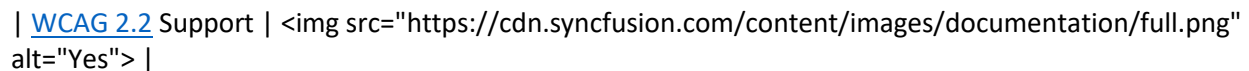
### Accessibility in ASP.NET MVC Sidebar component

The Sidebar component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Sidebar component is outlined below.

| Accessibility Criteria | Compatibility |

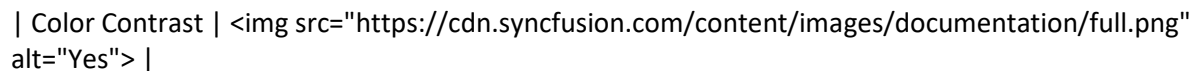
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

| Right-To-Left Support |  alt="Yes"> |

| Color Contrast |  alt="Yes"> |

| Mobile Device Support |  alt="Yes"> |

```

| Keyboard Navigation Support | |
| Accessibility Checker Validation | |
| Axe-core Accessibility Validation | |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

The Sidebar component followed the [WAI-ARIA](#) patterns to meet accessibility standards. By default, the Sidebar utilizes the [complementary](#) role, with the option to modify the ARIA role based on provided attributes to the root element, depending on the specific use case.

If there are multiple complementary landmark roles or aside elements in a document, it is important to provide a label for each landmark using the `aria-label` attribute. Alternatively, if the aside has a descriptive title, it can be referenced using the `aria-labelledby` attribute. This label will help assistive technology users quickly understand the purpose of each landmark.

For optimal accessibility, it is recommended to incorporate a trigger component that is navigable via a keyboard, such as a button. If this is not feasible, inclusion of the `tabIndex` attribute becomes necessary. Furthermore, explicit handling of the `aria-expanded` attribute is required for the trigger element.

### Keyboard interaction

The Sidebar component does not have any inbuilt keyboard interaction support. However, you can utilize the keyboard interactions of focusable elements within the Sidebar component.

### Ensuring accessibility

The Sidebar component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Sidebar component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Sidebar component with accessibility tools.



See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## How To

### Initialize the Sidebar with ListView

Any HTML element can be placed in the Sidebar content area. Sidebar supports all types of HTML structures like `TreeView`, `ListView`, etc.

In the following example, the Sidebar is rendered with `ListView` component in its content area.

- Add the HTML div tag with its id attribute as `default` in your `index.html` file to initialize the Sidebar.

## CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Type(Syncfusion.EJ2.Navigations.SidebarType.Over).Width("100%").Co
ntentTemplate(@<div>
 <div class="title1"> Menu </div>
 <div class="closebtn">
 <!-- Button element declaration -->
 @Html.EJS().Button("close").CssClass("e-btn close-
btn").Content("CLOSE SIDEBAR").IconCss("e-icons close-icon").Render()
 </div>
 <div id="listcontainer">
 <!-- Listview element declaration -->

@Html.EJS().ListView("list").DataSource((IEnumerable<object>)ViewBag.dataSou
rce).Render()
 </div>
 <div class="sub-title">
 * ListView component is placed inside the sidebar content area.
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- Main Content declaration -->
<div>
 <div class="title2">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- Button element declaration -->
 @Html.EJS().Button("toggle").Content("Toggle
Sidebar").IsToggle(true).CssClass("e-info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 // Create instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
```

```

 // Toggle button to open and close the sidebar
 document.getElementById('toggle').onclick = function () {
 defaultSidebar.toggle();
 }
 // Close the sidebar
 document.getElementById('close').onclick = function () {
 defaultSidebar.hide();
 }
 });
</script>
<style>
/* ListView element styles */
#listcontainer {
 width: 100%;
}
#list {
 margin: 0 auto;
 width: 30%;
}
.e-listview .e-list-item {
 text-align: center;
 font-size: 14px;
 padding: 0;
}
/* Button element styles */
.e-btn.close-btn :hover { /* csslint allow: adjoining-classes*/
 box-shadow: none;
 background: transparent;
}
.close-btn, .e-listview .e-list-item, #default-sidebar {
 background-color: rgb(20, 118, 210);
 color: #ffffff;
}
.close-icon::before {
 content: '\e945';
}
.close-btn {
 box-shadow: none;
}
.close-btn:hover {
 color: #fafafa;
}
.e-icons.close-icon { /* csslint allow: adjoining-classes*/
 line-height: 2.2;
}
.closebtn {
 top: 15px;
 line-height: 36px;
 height: 42px;
 color: black;
 position: absolute;
 right: 10px;
}
/* Sample level styles */
.title1 {
 text-align: center;
 font-size: 20px;
}

```

```

 padding: 15px;
 }
 .title2 {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 .center-align {
 text-align: center;
 padding: 20px;
 }
 body {
 margin: 0;
 }
</style>

```

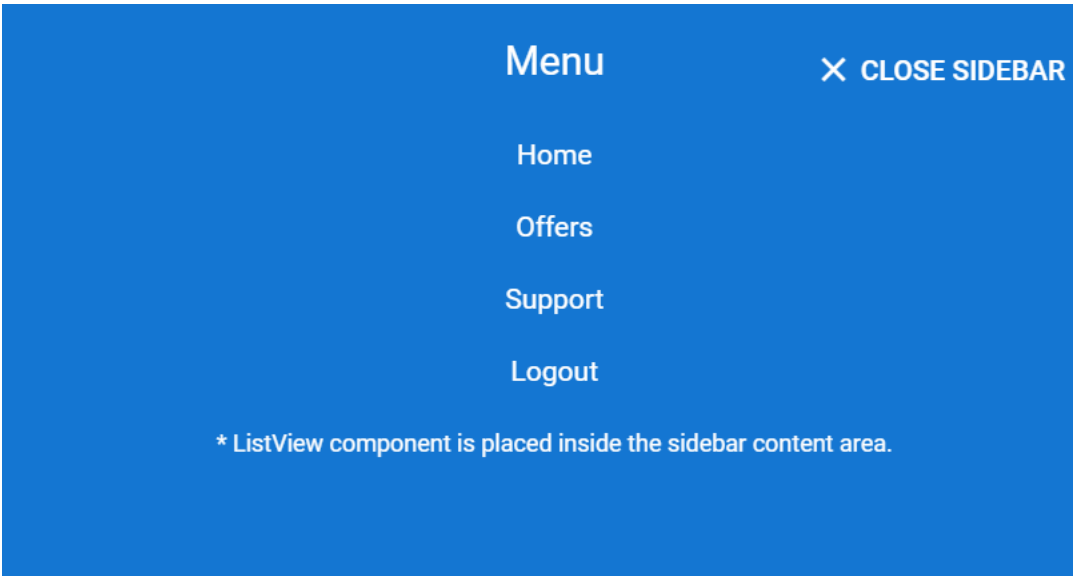
### SIDEBAR.CS

```

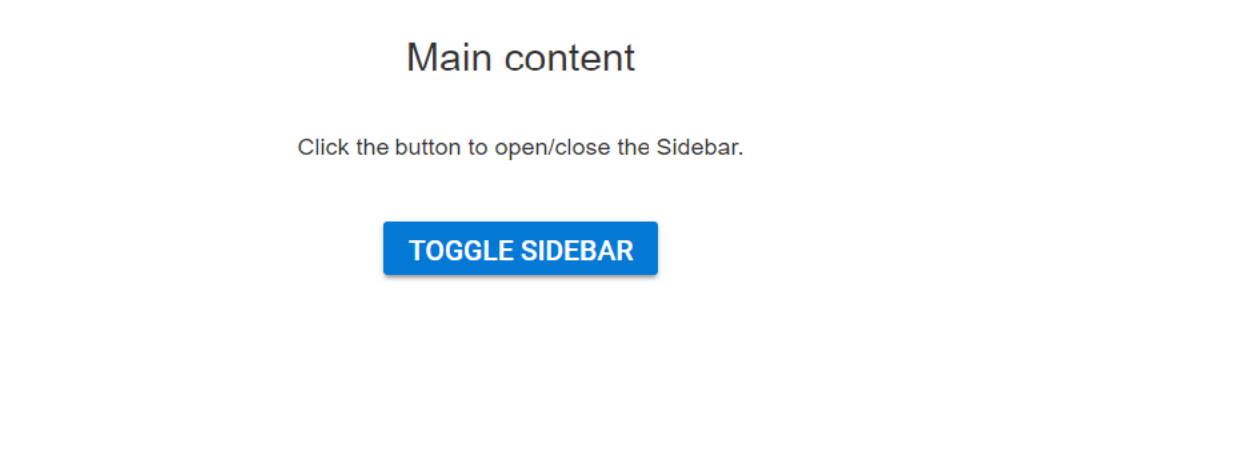
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult sidebar()
 {
 List<object> data = new List<object>();
 data.Add(new { text = "Home", id = "list-01" });
 data.Add(new { text = "Offers", id = "list-02" });
 data.Add(new { text = "Support", id = "list-03" });
 data.Add(new { text = "Logout", id = "list-04" });
 ViewBag.dataSource = data;
 return View();
 }
 }
}

```

Output be like the below in Expanded state, the sidebar width is set as 100%.



In Collapsed state, the output be like the below.



Open and close the Sidebar

Opening and closing the Sidebar can be achieved with built-in public methods.

| Method   | Description                                           |
|----------|-------------------------------------------------------|
| show()   | Method to open the Sidebar.                           |
| hide()   | Method to close the Sidebar.                          |
| toggle() | Method to toggle the open/close state of the Sidebar. |

In the following sample, toggle method has been used to show or hide the Sidebar on button click.

CSHTML

```
<!-- sidebar element declaration -->
```

```

@{Html.EJS().Sidebar("default-
sidebar").ShowBackdrop(false).Open("Open").Close("Close").ContentTemplate(@<
div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 Click the button to close the Sidebar.
 </div>
 <div class="center-align">
 <!-- Button element declaration -->
 @Html.EJS().Button("close").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- Main Content declaration -->
<div>
 <div class="title">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- Button element declaration -->
 @Html.EJS().Button("toggle").Content("Toggle Sidebar").CssClass("e-
info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 // Create instances for sidebar element
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 // Toggle button to open and close the sidebar
 document.getElementById('toggle').onclick = function () {
 defaultSidebar.toggle();
 }
 // Close the sidebar
 document.getElementById('close').onclick = function () {
 defaultSidebar.hide();
 }
 });
 function Open() {
 console.log("Sidebar Opened");
 }
 function Close() {
 console.log("Sidebar Closed");
 }
</script>
<style>
 /* Button element styles */
 #close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
 }
 /* sample level styles */
 .title {
 text-align: center;
 font-size: 20px;

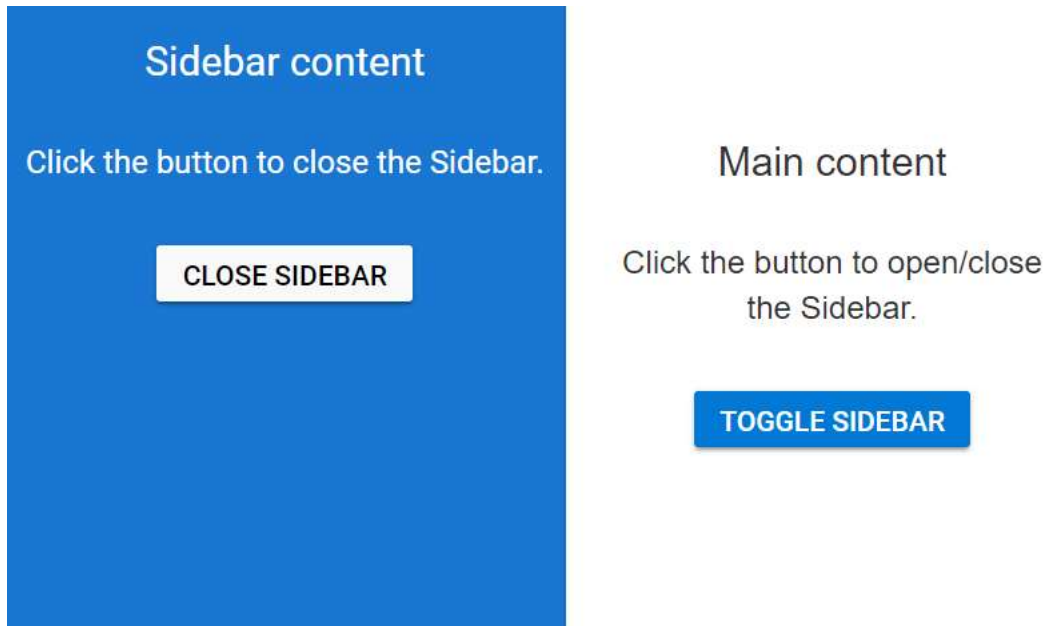
```

```
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 .center-align {
 text-align: center;
 padding: 20px;
 }
 body {
 margin: 0;
 }
 /* Sidebar element styles */
 #default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
</style>
```

### OPEN.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult open()
 {
 return View();
 }
 }
}
```

Output be like the below.



### Multiple Sidebar in SideBar Control

Two Sidebars can be initialized in a web page with same main content. Sidebars can be initialized on right side or left side of the main content using [position](#) property.

**Note:** The HTML element with class name `e-main-content` will be considered as the main content and both the Sidebars will behave as side content to this main content area of a web page.

In the following sample, more than one sidebar is rendered based on [position](#) property.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Type(Syncfusion.EJ2.Navigations.SidebarType.Push).Width("280px").C
ontentTemplate(@<div>
 <div class="title"> Left Sidebar content</div>
</div>).Render();
}
@{Html.EJS().Sidebar("default-
sidebar1").Type(Syncfusion.EJ2.Navigations.SidebarType.Push).Width("280px").
Position(Syncfusion.EJ2.Navigations.SidebarPosition.Right).ContentTemplate(@
<div>
 <div class="title"> Right Sidebar content</div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title">Main content</div>
 <div class="sub-title"> Click the button to open/close the
Sidebar.</div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("toggle").Content("Left Toggle
Sidebar").CssClass("e-info").Render()
```

```

 @Html.EJS().Button("toggle1").Content("Right Toggle
Sidebar").CssClass("e-info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 // Toggle button to close and open the sidebar
 document.getElementById('toggle').onclick = function () {
 var defaultSidebar = document.getElementById("default-
sidebar").ej2_instances[0];
 defaultSidebar.toggle();
 }
 // Close the sidebar
 document.getElementById('toggle1').onclick = function () {
 var defaultSidebar1 = document.getElementById("default-
sidebar1").ej2_instances[0];
 defaultSidebar1.toggle();
 }
 });
</script>
<style>
 /* sample level styles */
 .center-align {
 text-align: center;
 padding: 20px;
 }
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
 /* Button styles */
 #close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
 }
 /* sidebar element styles */
 #default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
 #default-sidebar1 {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
</style>

```

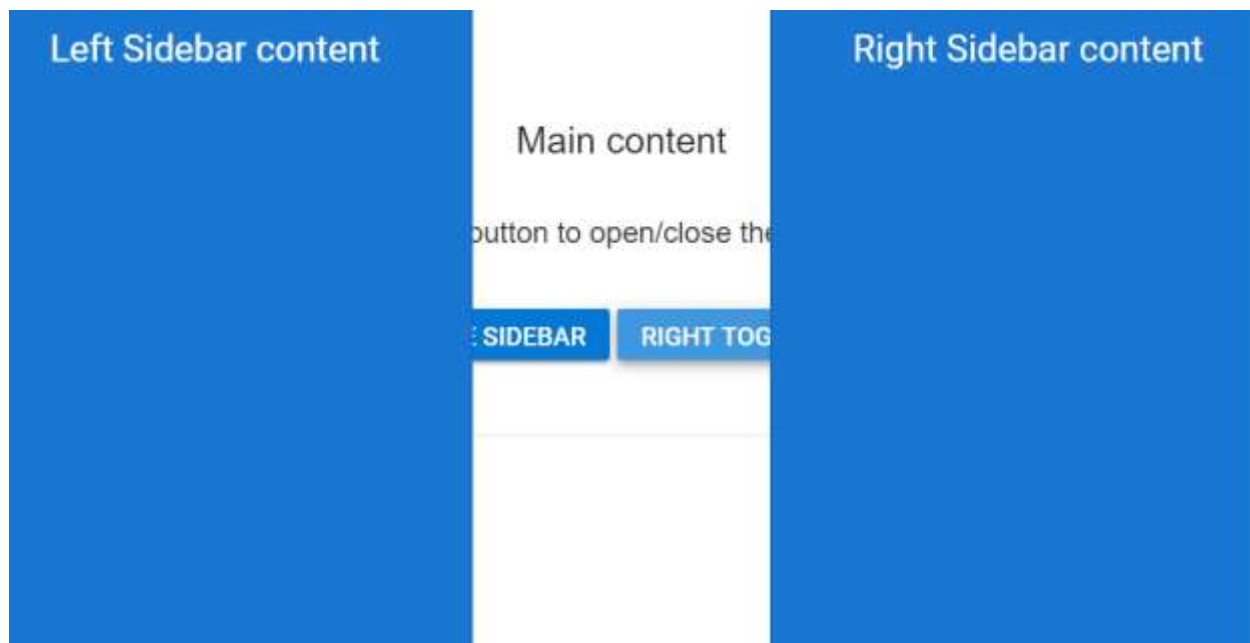
## MULTIPLE.CS

```
using System;
```



```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult multiple()
 {
 return View();
 }
 }
}
```

Output be like the below.



### Sidebar with fixed position

The Sidebar does not require any specific style to make it as a fixed one. By default, the Sidebar position will be in fixed state. The following example demonstrates that the Sidebar is rendered with a fixed position. The position of the Sidebar will not change when scrolling the main content area.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").CloseOnDocumentClick(false).Width("260px").ContentTemplate(
@<div>
 <div class="sidebar-header header-cover" style="background-
color:#0378d5">
 <div class="image-container">
 <div class="sidebar-image">
 </div>
 </div>
 </div>
 </div>
</div>
```

```

 <div style="padding: 0 0 5px 0;">

 john.doe@gmail.com

 </div>
 </div>
 <!-- Sidebar navigation -->
 <ul class="nav sidebar-nav">

 <i class="sf-icon-sidebar sf-icon-file"></i>
 Inbox

 <i class="sf-icon-sidebar sf-icon-starred"></i>
 Starred

 <i class="sf-icon-sidebar sf-icon-recent"></i>
 Snoozed

 <i class="sf-icon-sidebar sf-icon-important"></i>
 Important

 <i class="sf-icon-sidebar sf-icon-offline"></i>
 Sent

 <i class="sf-icon-sidebar sf-icon-backup"></i>
 Draft

</div>).Render();
}
<!-- end of sidebar element -->
<!-- Main Content declaration -->
<div>
 <div class="content">
 <div id="left">

 </div>
 <div id="center">
 Inbox

```

```

 </div>
 <div id="right">

 </div>
 </div>
 <div>
 <div class="e-control e-listview e-list-template e-touch">
 <ul class="e-list-parent e-ul ">
 <li class="e-list-group-item e-level-1" role="group" data-
uid="group-list-item-Today"
 aria-level="1">
 <div class="e-text-content" role="presentation"><span
class style="width: 100%; margin-left: 2%; margin-top: -
2%;">Today</div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>
 Albert
Lives
 </div>
 <span class="received e-list-content e-second-
heading">Opening for Sales Manager

 Hello Uta
Morgan,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>

 Ila Russo

 </div>
 <span class="received e-list-content e-second-
heading">
 Business dinner invitation

 Hello Jelani
Moreno,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">

```

```

 <div>

 Garth Owen

 </div>
 <span class="received e-list-content e-second-
heading">Application for Job Title

 Hello Ila
Russo,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>
 Ursula
Patterson
 <span class="received e-list-content e-second-
heading">Hello Kerry Best,

 Programmer Position
Application

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>

 Nichole Rivas

 </div>
 <span class="received e-list-content e-second-
heading">Annual Conference

 Hi Igor Mccoy,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>

 Nichole Rivas

 </div>
 </div>


```

```

 <span class="received e-list-content e-second-
heading">Annual Conference

 Hi Igor Mccoy,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>

 Nichole Rivas

 </div>
 <span class="received e-list-content e-second-
heading">Annual Conference

 Hi Igor Mccoy,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>

 Nichole Rivas

 </div>
 <span class="received e-list-content e-second-
heading">Annual Conference

 Hi Igor Mccoy,

 </div>

 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>

 Nichole Rivas

 </div>
 <span class="received e-list-content e-second-
heading">Annual Conference

 Hi Igor Mccoy,

 </div>


```

```


 <li class="e-list-item">
 <div class="e-list-wrapper e-list-avatar e-list-multi-
line">
 <span class="e-avatar e-icon sf-icon-
profile">
 <div>
 Ursula
Patterson
 </div>
 <span class="received e-list-content e-second-
heading">Hello Kerry Best,

 Programmer Position
Application

 </div>

 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 var sidebarTreeview = document.getElementById("default-
sidebar").ej2_instances[0];
 // Toggle the Sidebar
 document.getElementById('hamburger').onclick = function () {
 sidebarTreeview.toggle();
 }
 });
 //create instances for sidebar element
</script>
<style>
 .control-section {
 overflow: hidden;
 }
 .e-listview .e-list-parent {
 margin: 0;
 padding: 15px 0 15px 0;
 }
 .e-listview .e-list-group-item {
 background-color: #ffffff;
 }
 .e-second-heading {
 color: rgba(0, 0, 0, 0.64);
 display: block;
 font-size: 15px;
 font-weight: 500;
 }
 .e-listview.e-list-template .e-list-wrapper.e-list-multi-line.e-list-
avatar .e-avatar {
 top: 1.5666em;
 border-radius: 50%;
 border-color: #4f4b4b;
 border: 1px solid #BADA55;
 }

```

```
.e-listview {
 overflow: hidden;
}
.sf-icon-profile:before {
 float: right;
 content: '\e717';
 font-size: 25px;
}
.e-avatar {
 background: none;
 color: #000;
}
.e-today {
 height: 50px;
 padding: 15px;
}
 .e-today span {
 border-color: rgba(0, 0, 0, 0.12);
 color: rgba(0, 0, 0, 0.54);
 font-size: 15px;
 font-weight: 600;
 height: 36px;
 }
.content {
 height: 47px;
 background: #0378d5;
 width: 100%;
}
#left {
 float: left;
 width: 30px;
 height: 47px;
}
#center {
 display: inline-block;
 margin: 0 auto;
 width: 0px;
 height: 47px;
 padding: 9px 0 12px 0;
 font-size: 18px;
 color: #ffff;
 float: left;
 text-align: center;
 padding-left: 40px;
}
#right {
 float: right;
 width: 28px;
 height: 47px;
 padding: 16px 0 0 0;
 color: #ffff;
}
.sf-icon-file:before {
 content: '\e701';
}
.sf-icon-share:before {
 content: '\e70b';
}
```

```
}
.sf-icon-starred:before {
 content: '\e708';
}
.sf-icon-star:before {
 content: '\e707';
 font-size: 15px;
 color: #000;
 float: right;
}
.sf-icon-recent:before {
 content: '\e70c';
}
.sf-icon-important:before {
 content: '\e713';
}
.e-sidebar.e-left {
 border: none;
 left: inherit;
}
.sf-icon-offline:before {
 content: '\e714';
 font-size: 30px;
}
.sf-icon-backup:before {
 content: '\e712';
}
.sf-icon-down:before {
 content: '\e709';
}
.sf-icon-search:before {
 content: '\e711';
 font-size: 18px;
}
.sf-icon-sidebar {
 content: '\e701';
 color: #000000;
 display: inline-block;
 margin-right: 16px;
 width: 40px;
 text-align: left;
 font-size: 20px;
 vertical-align: middle;
}
ul.nav.sidebar-nav {
 padding: 5px 0 5px 0;
 margin: 0px;
 max-height: 477px;
 overflow-y: auto;
}
.sidebar-nav {
 list-style: none;
 padding: 5px 0 5px 0;
}
a.sidebar-brand {
 color: #ffffff;
 font-size: 15px;
```



```

 text-decoration: none;
 }
 .nav.sidebar-nav li:hover {
 text-decoration: none;
 background-color: #eee;
 }
 span.sf-icon-down.icon {
 font-size: 12px;
 }
 .sidebar-nav li a {
 position: relative;
 cursor: pointer;
 user-select: none;
 display: block;
 height: 48px;
 line-height: 48px;
 width: 241px;
 padding: 0 56px 0 20px;
 text-decoration: none;
 font-weight: 500;
 overflow: hidden;
 font-size: 14px;
 }
 /* sample-level styles */
 .image-container {
 padding: 18px 0 0 16px;
 }
 .sidebar-image {
 background:
url(https://ej2.syncfusion.com/showcase/angular/expensetracker/assets/images
/user.svg) no-repeat scroll 0 0 transparent;
 background-position-x: left;
 height: 85px;
 width: 100%;
 background-size: 85px;
 /* margin-left: 20px; */
 }
 .sidebar-brand {
 height: 48px;
 line-height: 48px;
 padding-left: 20px;
 padding-right: 40px;
 text-decoration: none;
 font-weight: 500;
 overflow: hidden;
 display: inline-flex;
 }
 .sidebar-image img {
 width: 54px;
 height: 54px;
 margin: 16px;
 border-radius: 50%;
 -webkit-transition: all 0.2s ease-in-out;
 -o-transition: all 0.2s ease-in-out;
 transition: all 0.2s ease-in-out;
 }
 #default-sidebar {

```

```

 color: #ffffff;
 height: 477px;
 border-right: 0.5px solid #80808070;
 border-bottom: 0.5px solid #80808070;
 }
 /* sample-level styles */
 #hamburger.menu {
 font-size: 23px;
 cursor: pointer;
 float: left;
 line-height: 50px;
 position: absolute;
 z-index: 1000;
 margin-left: 25px;
 color: #ffff;
 }

 #hamburger.menu:before {
 content: '\e10d';
 }
 #hamburger.menu.e-rtl {
 position: relative;
 float: right;
 }
}

@@font-face {
 font-family: 'e-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjciQ6oAAAEoAAAAVmNtYXBH1Ec8AAABsAAAAHJnbHlmKcX
fOQAAAKAAAAg4aGVhZBLt+DYAADQAAAAANmhoZWEHogNsAAAArAAAACRobXR4LvgAAAAAYAAAAA
wbG9jYQYkCgIAAAIkAAAAAGmlheHABGQEOAAABCAAAACBuYW1lR4040wAACngAAAJtcG9zdEFgIbw
AAAzAAAArAABAAADUv9qAFoEAAAA//UD8wABAAAAAAAAAAAAAAAAADAABAAAAQAAlbrm7l8
PPPUACwPoAAAAANfuWa8AAAAA1+5ZrwAAAAAD8wPzAAAAACAACAAAAAAAAAAAAEAAAAQAIAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPPqAZAABQAAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4QLhkANS/2oAWgPzAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAAAAAAAgAAAAAAMAAAAQA
AABQABABeAAAADgAIAAIABuEC4QnhD+ES4RvhkP//AADhAuEJ4QvhEuEa4ZD//wAAAAAAAAAAAA
AAAABAA4ADgAOABYAFgAYAAAAQACAAYABAADAAGABWAKAAKABQALAAAAAAAAAB4AQABaAQYB5gJ
kAnoCjgKwA8oEHAaaaaIAAAAA+oDlQAEAAoAAAEFESERCQEVCQE1AgcBZv0mAXQB5P4c/g4Cw/D
+lwFpAcP+s24BTf6qbgAAAAEAAAAAAAA+oD6gALAAATCQEXCQEHcQEnCQF4AYgBiGP+eAGIY/54/nh
jAYj+eAPr/ngBiGP+eP54YwGI/nhjAYgBiAAAAwAAAAAD6gOkAAMABwALAAA3IRUhESEVIREhFSE
VA9b8KgPW/CoDl1vwq6I0B64wB640AAAEAAAAAAAA+oD4QCaAAABMx8ahQEpdjEPah8bIT8bNS8SPxs
CAA0aGhgMDAsLCwoKCgkJCQgHBwYGBgUEBAMCAgECAwUFBggICQoLCwwMDg0GAgEBAgIDBAMIBiI
dHh0cHB0ZFhUSEAcFBgQDAwEB/CoBAQMDBAUGBw8SFRYYGhsbHB0cHwsJBQQEAWIBAQMEDg0NDAs
LCQkJBwYGBAMCAQEBAgIDBAQFBQYGBwgICakJCgoKCwsLDAwMGRoD4gMEBwQFBQYGBwgICakKCgs
LDAwNDQ4ODxAQEBEWFxYWFhYVFRQUExIRERAOFxMLCggIBgYFBgQMDAwNDg4QDxERERIJCQkKCQk
JFRQJCQoJCQgJEhERERAPDw4NDQsMBwgFBgYICQkKDAwODw8RERMTEUUFhUWFxYWFxEQEBAPDg4
NDQwMCwsKCgkICAgHBgYFBQQEBAQAAAAAAwAAAAAD8wPzAEEAZQDFAAABMx8FFREzHwYdAg8GIS8
GPQI/BjMlKwEvBT0CPwUzNzMfBR0CDwUrAi8FPQI/BTMnDw8fFz8XLxcPBgI+BQQDAwMCAT8EBAM
DAWIABAQIDAwmEBP7cBAQDAwMCAQECAwMDBAQ/PwQEAWMDAgEBAgMDAwQE0AUEAWMDAgEBAgMDAwQ
FfaUEAWMDAgEBAgMDAwQFvRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRocHR4eHyAgISI
iISAgHx4eHRsbGRcWFRMREA4LCQgFAwEBawUHCgsOEBETFRYXGRsbHR4eHyAgISIiISAgHx4eAqY
BAGIDBAQE/rMBAQEDAwQEBGgEBAQDAgIBAQEBAgIDBAQEaAQEBAMDAQEBA0AECawMDBAVoBAQDAwM
CAeUBAgIEAwQEaAUEAwMDAgEBAgMDAwQFaAQEAwQCAgElERMVfhcZGhwdHh4fICAhIiIhICAfHh4
dGxsZFxyVExEQDgsJCAUDAQEDBQcKCw4QERMVfhcZGxsDhH4fICAhIiIhICAfHh4dHBoZFxyVExE
QDgsKBwUDAQEDBQcKCw4AAAAIAAAAAA9MD6QALAE8AAAEAOAQcuASc+ATceAQEHBgcNjgYPAQYWHwE
GFBChDgEfAR4BPwEWHwEeATsBMjY/ATY3Fxy2PwE2Ji8BNjQnNz4BLwEuAQ8BJi8BLgErASIGaps
BY0tKYwICY0pLY/7WEy4nfAkRBWQEAWdqAwNqBwMEZAURCXwnLhMBDgnICg4BEy4mfQkRBGQFAwh
pAwNpCAMFZAQSCH0mLhMBDgrICQ4B9UpjAgJjSkpjAgJjAZWEFB4yBAYIrggSBlIYmhhSbHIrgg

```

```

FAzIfE4QJDAwJhBQeMgQGCK4IEgZSGDIYUgYSCK4IBQMyHxOECQwMAAEAAAAAAwED6gAFAAAJAic
JAQEbAef+FhoBzf4zA+v+Ff4VhWHMac0AAAAAAQAAAAADAQPqAAUAAAEXCQEHAQLlHf4zAc0a/hY
D6x7+M/40HwHrAAEAAAAAA/MD8wALAAATCQEXCQE3CQEnCQENAY7+cmQBjwGPZP5yAY5k/nH+cQO
P/nH+cWQBjv5yZAGPAY9k/nEBjwAAAwAAAAAD8wPzAEAAgQEBAAA1Dw4rAS8dPQE/DgUVDw4BPw4
7AR8dBRUfHTsBPx09AS8dKwEPHQL1DQ00Dg4PDw8QEBAQERERERUUFbQTExITEREREBAPDw00DAw
LCwkJCACGBgQEAgIBAgIEAwUFBgYHBwkICQoCygECAGQDBQUGBgCHCQgJCv3QDQ00Dg4PDw8QEBA
QERERERUUFbQTExITEREREBAPDw00DAwLCwkJCACGBgQEAgL8fgIDBQUHCAkKCwNDg8PERESExQ
UFRYWFhgXGBkZGRoaGRkZGBcYFhYWFQRUEXIREQ8PDg0MCwoJCACFBQMCAGMFBQCICQoLDA00Dw8
RERITFBQVfYhYWGBCYGRkZGhoZGRkYFhgWfYhYVFBQTEhERDw8ODQwLCgkIBwUFAwLFCgkICQCHBgY
FBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehMTFBQUFREREREQEBAQDw8PDg4ODQ31ERE
RERAQEBAAPDw8ODg4NDQIwCgkICQCHBgYFBQMEAgIBAgIEBAYGBwgJCQsLDAwODQ8PEBARERETehM
TFBQUFRoZGRkYFhgWfYhYVFBQTEhERDw8ODQwLCgkIBwUFAwICAwUFBwgJCgsmDQ4PDxEREhMUFBU
WFhYFhgZGRkaGhkZGRgXGBYWFhUUFBMSEREPdW4NDAsKCQgHBQUdAgIDBQUHCAkKCwNDg8PERE
SExQUFRYWFhgXGBkZGQAAQAAAAAD6gPqAEMAABMHw8RDw8hLw8RPw6aAswNDgwMDAsKCgGIBwU
FAwIBAQIDBQUHCAgKCgsmDAwODf00DQ4MDAwLCgoICACFBQMCAGQECawUFBwgICgoLDAwMDgPrAQI
DBQUHCAgKCgsmLDA0NDv00Dg0NDAsLCgoICACFBQMCAGQECawUFBwgICgoLCwNDQ4CzA4NDQwLCwo
KCAGHBQUdAgAAABIA3gABAAAAAAAEAAAAABAAAAABAA0AAQABAAAAAAACAAcADgABAAAAAA
DAA0AFQABAAAAAAAEAA0AIgABAAAAAAFAAsALwABAAAAAAAGAA0AOgABAAAAAAAKACwArWABAAA
AAAAALABIAcWADAAEECQAAAAIAhQADAAEECQABABoAhWADAAEECQACAA4AoQADAAEECQADABoArWA
DAAEECQAEABoAyQADAAEECQAFABYA4wADAAEECQAGABoA+QADAAEECQAKAFgBEWADAAEECQALACQ
BayB1LW1ljB25zLW1ldHJvUmVndWxhcmtUtaWNvbnMtbWV0cm91LW1ljB25zLW1ldHJvVmVyc2lvbiA
xLjB1LW1ljB25zLW1ldHJvRm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z1c2lvbiBNZXRybyBTdHV
kaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgBvAFIAZQBnAHU
AbABhAHIAZQAtAGkAYwBvAG4AcwAtAG0AZQB0AHIAbWBlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgB
vAFYAZQByAHMAaQBvAG4AIAAxAAC4AMABlAC0AaQBjAG8AbgBzAC0AbQB1AHQAcgBvAEYAbwBuAHQ
AIABnAGUAbgB1AHIAZQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAAB
NAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8
AbQAAAAACAAAAAAAOAAAAAAAEAAAAAAAwBAgEDAQQBBQEQAQcBCAEJAQo
BCwEMAQ0AB2hvbWUtMDELQ2xvc2UtaWNvbnMhbnVudS0wMQRlc2VyB0JUX2luZm8PU2V0dGluZ19
BbmRyb2lkDWNoZXZyb24tcmlnaHQMY2hlbnJvbi1sZWZ0CE1UX0NsZWfYDE1UX0p1bmttYWlscwR
zdG9wAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
@@font-face {
 font-family: 'e-sb-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgT1MvMj0gSSgAAAEoAAAVmNtYXNt+g6AAAB6AAAAGZnbHlmMmV
udQAAaogAAAZ0aGVhZBMA8GsAAADQAAANmhoZWEHkaOAAAArAAAACRobXR4Zan/8gAAAYAAAAB
obG9jYSiaJP4AAAQJAAAAANm1heHABKwBwAAABCAAAACBuYw1lTGtTDAAAD3wAAAJJcG9zdLRM+38
AABHIAAABggABAAADUv9qAFoEAP/0//sD7QABAAAAAAGgABAAAAQAAoeBeD18
PPPUACwPoAAAAANgAlc4AAAAA2ADVzv/0AAD7QPqAAACAAACAAAAAAAEAAAAaAGQABwAAAA
AAgAAAAoACgAAAP8AAAAAAQAAPAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAUUGZFZABA5wDnGANS/2oAWgPqAJYAAAAABAAAAAABAAAAAPo//Q
D6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+j//gP
oAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAD6AAAA+gAAAPoAAAAAACAAAAwAAABQAAwABAAA
AFAAEAFIAAAEAQAQA5xj//wAA5wD//wAAAAEABAAAAEAAGADAAQABQAGAAcACAAJAAoACwA
MAA0ADgAPABAAEQASABMAFAAVABYAFwAZABgAAAAAAAVgCIANABBAE+AUWBYgGMAgBtgHOAmA
CsALUA24D2gRWBjAE0gVCBZgFuAXsBkwGegAAAP/9AAAA+oDeQASACQALwAAEWYsfY2NxcTBRC
OAScmAjqCOATCRHqEXISc3NTYnAQYiJwEOATCTFjI3ATc2NSEWBhPrxoDeTV0+/mxgQKlhorsQBw1
lDB4Rabdq+QEN/tSHFAf+2wUEmv8HEQCBAQC/dUBAonK/tYdD15cRwGwokpMSwsXAQi3GDGR/rg
SJRBRZNoNCP7bBgYBJQIMmf8ABwcBAACeBAoAAAAADAAAAAPqA20ABAAJABkAAAKBESERJRUJATU
jER4BMYEYnJURNcYnIQ4BAfYBtvyVA2v+Sv5LPwEjGwNrGyMjG/yVGyMBdwFF/gICAHEl/rsBRYp
9jxsjIxsCcRsjaQEjAAAAwAAAAAD6gNtABQAGQApAAABJyYOARYfARUhNTc+AS4BDwERCQE1FQk
BNSMRHgEzITI2NRE0JichDgEDrNILGBAEC/b8lekLBBAYC8UBrQG+/kL+Uz8BIxsDaxsjI xv81Rs
jATaYCAQWQGiyKzWoCBkWBAiOAXf+wAFLBr/+tQFBKf2PGyMjGwJxGyMBASMBQAAAAAD7QNLAQ
ABwAKAA0AFgAANyElByc3BRMBJQkCISUzNzYmByIGJwYDsP5tRUWZAYED/FABff5/AdwB2PxQAv6

```

4LAYRCwcIAaLaMjIE0gHu/hLSARz+uwFbGVgZEAECAGAAAAFAAAAAAPqA64ABQAIaAsAEQAZAAA  
1ISUXPWefLQENARElASCHASUFESERNyc1JQN6/QUBJAefXQGl/u0BE/3I/s4DX/7EX17+ugGg/gA  
D5wIC/hp+xgEFQOHF/vrRAeku/ttDQAEqrI/9ZAJ1AQE01AAAAEAAAAAA+oDKWACAAA3IQECA+j  
+DMICaQAAAAACAAAAAPqAxYAAgAFAAABIQkBIQEDYf0rAWr+DAPo/gwBFQGi/iACPwAAAAIAAAA  
AA+oDuwAJABMAAAezBxcnBzcnMzcHIQUdJQUdJSEDAj/kuUi8u0i55Eh0/oEBNXUBMwE0dQE1/oF  
1AiWB04KC04HULtf+pdBWAVvXAVkAAAEAAAAAA+oD6gAJAAABBRMDJQUDEyUDAVz+pvo6ATUBNTz  
6/qebAqE1/wD+lqurAWoBADUBSQAAAQAAAAAD6gMrAAIAACUBIQH2AfT8GMICaQAAAAIAAAAA+o  
DiAADAAsAADchEyE3FSMVITUHNUIDaT/8GKdPayn+tmUCda5ERUVEAAUAAAAAA+ID6gAJABUALQA  
5AFsAAAEeARcVITU+AT8BDgEHLgEnPgE3HgEFHgEFAQ4BBxUhNS4BJyM3PgE3LgEnDgEnDgEHLgE  
nPgE3HgEFFBYfAQcOAcVITUhNT4BNzMyFhc3Ji8BNz4BNy4BJw4BASxcwL91wJ7XMUBTzo7Twe  
BTzs6T/6uATQsAmaDAwKkAoNmAgIsNAECcVVVchwBPzAvPwICPy8wP/7kKSMDA1t1AgFy/swDgF8  
fGDwaGxUWCAYgJQECYkpJYgE2A3pCHR1cegPnO08BAU870k8BAU86N1obARKXaltbapcSARTaNLV  
yAgJxyy9AAQFALzA/AQE/MC1LGAEBHJjgdz06XHWDBgw3CgUCBBhIK0piAgJiAADAAAAAAPqA+o  
AFQAhAC0AAAEjDgEHHgEXPgE3NTM1IzUuAScjNSMBDgEHLgEnPgE3HgEFFgAXNgA3JgAnBgAB5AM  
PEAEBHxURFwesrwIMCAMLAcgF97q59wUF97m69/xbBgEb09QBGwUF/uXU0/71AiwiGhAWHwEBEAw  
DKAMIDQTu/uC59wUF97m69wUF97rT/uUGBgEb09QBGwUF/uUAGAAAAADugPpAAcAEgAANYERiXU  
hNSMTFzM3ETMRfzM3ATADikv9DUyeAmuRTpJqAv7bAQFLpqYBeQKR/fUCC5ECASQAAAAAAv/+AAA  
D6gMgADEAYwAAQYHDgEHBiMGBw4BBwYXFhcWNYEWNzY3Njc1JyYnLgEnLgE1NicuAScmBwYnJic  
mIyInBg8BJg4CFwcOARcWFxYzNh3JyY3PgI3Mjc2Nz4BNzYXFh8BPgE3Jy4BJyYnIyIB+TAiGBo  
FAQQhHRkiBwkNETschQIWHhgeCAMBAQIFCycbBgMCERVAKiYiBAMNDy5AFhdXLQQtV1IiAhA9MAC  
HLCUwJycGAYMPCCU5IAUBAwQLKR9IVD4oCREjEgIMPzUpKgYWAnoHIhg7IQUCDgsnGyopPRUKAQE  
QFyUMDBEFERAAhWYCAwYkICYkAgERAwUQDyycFk4FFgI5UTIDEVQzNiIeAQEBBT1EHY4gBwYMDCI  
4Ey0RDDELCAUBCzhSGBIBAAACAAAAAPqA4sACQBFAAA1JwcXNycHESMRAY4BBw4BBw4BBx4BFzM  
3Iy4BJz4BNzU3JjY3NhyXMDUzPgE3HgEXFR4BFAYHIxUzPgE3LgEnMS4BJw4BAfVJLZWULEk/Whx  
JJyo2CkhYAgJ5X3wBekJZAgFSRwUCGCckOgPEA2BDRVcBSFZRR0RhSW8CAVZCA4Ber3LZSSyUlCx  
JAVv+pQIrFhAKCjssE2pIVXQFPgJRPTxRagIBG0EMCB0fAUdUAQNcOjgCVH9MAT4Cc1ZiAhlfEQI  
BSwAAAAFAAAAAAPqA+kACwAXADIAPgBKAAABFwcXNxc3JzcnBycFDgEHLgEnPgE3HgElBycmIyI  
GBxc2PwEHDgEHHgEXPgE3LgEnIgYlFz4BMhYXNy4BIgYnFz4BIBYXNy4BIAYCOG5uLG5vLG9vLG9  
uAUgCfFxdFAICfFlcfP6AAwYVFi5UIS0rPacFFxkBBJ54d54DA553MFX+Xi06kqSSoIlDqr2p7C1  
c6QEE6VwtZv/+4v8Bim5uLG5uLG5vLG9vml17AgJ7XV17AwN7hAIBBSQhLcSGAQkiTyx3nwMDn3d  
3nwMDUCs7Pz87K0VISGgrXmNjXitobGwAAIAAAAA+oD6gALAB8AAAEQAQcuASc+ATceAQeARc  
yNj8BATcBMT4BNy4BJw4BAN4Donl6ogMDonp5ov2HBMWVQUuAwF3LP6IIycBBMWULcUCJnqiAwO  
ienmiAwOiezXFAY0oAv56LAGILW0+lMUDA8UABgAAAAADbQPqAAIABgAKAA0AFgAkAAAlNyc3Fzc  
nNxc3JzcnNscVHgEXMxEhESMRHgEXIT4BNxEnISIGAR9sSCRIs0gkSEhIg3Y/ASgbl/2PPgEnGwJ  
xGh8B6P49GyeiJEGkSLRHJEhISL9wLJ0bIgH9cANr/JUBIwEBIXsCvO0jAAAABgAAAAAD6wPpAAM  
ABwAZACsALgBFAAA1MzUjNTM1IzcxFG4CIyEiLgI/AT4BMhYnBwYeAhchPgMvAS4BIgYTBSUnER4  
BFyE1IREBFjI3AREzES4BJyEOAQKxPz8/P3GACwIYKhN+/xkpGQEKgQwrNivQgRECKkUqAQEQRSO  
DEoEUSVlIr/69/r2CAREOAZn+hgF0CBQIAXQ/AREO/LYOEW+PtsD+hUrKBcXKCsV+hgZGgb6JEt  
FJgEBJkVLJPonKysBrOPjH/2wDhEBPwH1/voFBQEG/qMBmg0RAQERAAADAAAAAPqAzAAGQAuADk  
AAAEVFCsBiH0BFDsBMh0BFj8BNjQvATQnKwEGJREeATMhNTQ2OwE1NCcBBiInAQ4BNxcWMj8BNjc  
hFBYDBAR6CQn6CgEF3gMD3gECAGH8/gEMCQGqDAqMcF7tCBMI/uoHBjL0BRAI8QUB/fICAmJ9Cgm  
DCgl6CQbLAWcCzAIBAYv+mwkNgAoP3g8E/u0HBwEWAgst7gCH8QQIBACAAwAAAAAD6gNtAAQABwA  
LAAAlIREJAScJAQMhESEDpPyVAbUBt1T+nv6ekgPo/Bi+AmD+vAFFEP74AQj9UQLuAAcAAAAAA+o  
CwADAAGADAAQABMAFgAZAAATMzUjBSElByclMzUjNTM1IwUXEQE3JwUBIQLIyAESAs/+mVm/ez  
IyMjIAvfx/Sry8gFoAwf9MQGBIXycUVESIUwhnZMBU/6tk8D3AR8ABAAAAAADzgPqAA8AIAAvADs  
AACUjJzcjFwcjAwYCFyEuAQcBHgEXPgE3LgEnNDY1JiMOASUzMhYdARQgKwEVLwE3FQceArc+ATc  
uAScOAQJeJjsvZDI2Lj+tjgMDryP1EP5EA5Rvb5MFY38CAxMTb5QCA3QKDAwKdzs4d5ACWUFDWAE  
CWUFDWCuAAV1SKARm/v4Rv4sEASJxkNICKm8FgmAJEACDA5OiDAooCgwpLChMIy9DVwICWUFDWAI  
CFAAAAGAAAAAD3QPqAA0AGQAANxUhNS4BJw4BIiYnDgETHgEXPgE3LgEnDgEQa84DknEvc39zL3G  
S1AOac3SaAwOadHOahIKCdaMRJCgoJBGjAef0mgMDmnR0mgICmgAAABIA3gABAAAAAFAAAAAA  
BAAAAAABAAoAAQABAAAAAACAACwABAAAAAADAaOAEgABAAAAAEEAAoAHAABAAAAAFAAAs  
AJgABAAAAAAGAAoAMQABAAAAAAKACwAOwABAAAAAALABIAZwADAAEECQAAAAIAEQADAAEECQA  
BABQAewADAAEECQACAA4AjwADAAEECQADABQANQADAAEECQAEABQAsQADAAEECQAFABYAXQADAAE  
ECQAGABQA2wADAAEECQAKAFgA7wADAAEECQALACQBryB1LXNiLWljb25zUmVndWxhcmtUc2ItaWN  
vbnN1LXNiLWljb25zVmVyc2lvbiAxLjB1LXNiLWljb25zRm9udCBnZW51cmF0ZWQgdXNpbmcgU31  
uY2Z1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABlAC0AcwBiAC0AaQBjAG8  
AbgBzAFIAZQBnAHUAbABhAHIAZQAtAHMAYgAtAGkAYwBvAG4AcwBlAC0AcwBiAC0AaQBjAG8AbgB

```

zAFYAZQByAHMAaQBvAG4AIAAxAC4AMABlAC0AcwBiAC0AaQBjAG8AbgBzAEYAbwBuAHQAIABnAGU
AbgBlAHIAIAYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdAB
yAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAA
CAAAAAAAAAAaAAAAAAAAAAAAAAAAAAAAAAAAAABaBAGEDAQQBBQEGAQcBCAEJAQoBCwEMAQ0
BDgEPARABEQESARMBFAEVARYBFwEYARKBGgEbAAltYWlsLXNlbnQLaW5ib3gtMDIt2YLaW5ib3gt
tMDIt2YFaW5ib3gtPb3Blbi1tZXNzYWdlLXdmDGFycm93aGVhZC0wMRBhcnJvd2hlYWQtdG9wLXd
mCXJhdGluZy13ZgtYXRpbmctLS0wMxhcnJvd2hlYWQtdG93bi0wMQlmb2xkZXItMDMIdXNlcnM
td2YIY2xvY2stMDIGdXBsb2FkCG9uZWRYaXZlEWNSb3VklWRvd25sb2FkLXdmD3dvcmtb2ZmbG1
uZS13ZglzZWYyY2gt2Ypbn90ZS1tZW1vLTAxLXdmDGltcG9ydGFudC13ZglzZW50LW1haWwIaW5
ib3gt2YlbnWFpbC0tLXNlbnQJdXNlciliYWNrDHVzZXItcHJvZmlsZQAAAAA=)
format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"],
[class*=" sf-icon-"] {
 font-family: 'e-sb-icons';
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
span.e-text {
 color: #000000;
}
.title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
}
.sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
}
.center {
 text-align: center;
 display: none;
 font-size: 13px;
 font-weight: 400;
 margin-top: 20px;
}
body {
 padding-top: 0px;
 overflow: hidden;
}
[class^="sf-icon-"],
[class*=" sf-icon-"] {
 font-family: 'e-sb-icons';
 font-size: 15px;
 font-style: normal;
 font-weight: normal;
}

```

```

 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 }
 /* end of newTab support */
 .e-content-animation li.e-list-item {
 border-bottom: 0.5px solid #80808070;
 cursor: pointer;
 }
 .e-content-animation {
 border-bottom: 0.5px solid #80808070;
 height: 477px;
 overflow: auto;
 }
}
</style>

```

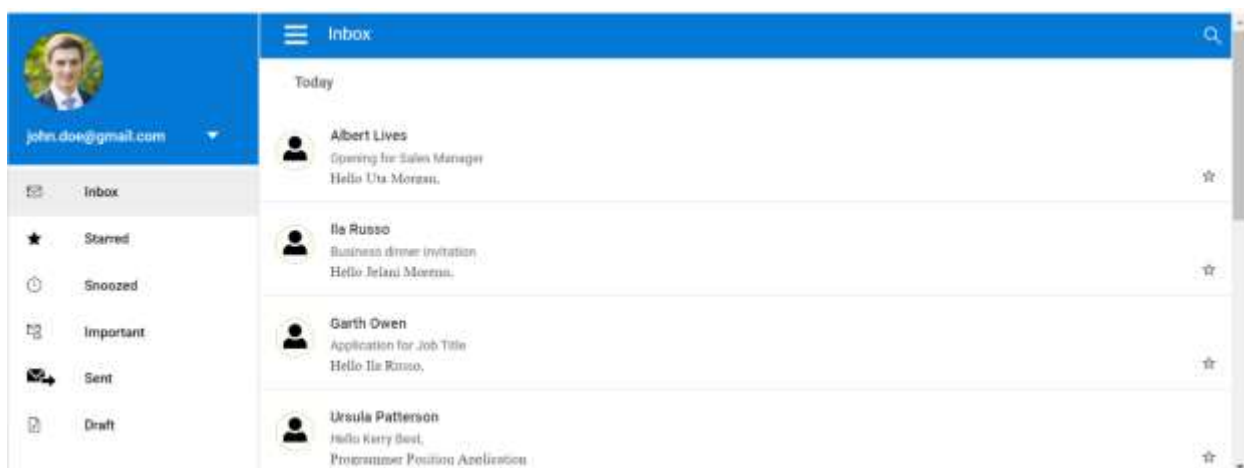
### FIXED\_POSITION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult fixed_position()
 {
 return View();
 }
 }
}

```

Output be like the below.



### Custom animation effects with sidebar

In the following example, the sidebar is rendered with custom animation effects. Click the buttons available in the main content area to check how the custom animations works with sidebar.

Sidebar will automatically adjust expanding animation to match any custom size specified in CSS styles.

#### CSHTML

```

<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("sidebar-
element").ShowBackdrop(true).Width("280px").ContentTemplate(@<div>
 <div class="title"> Sidebar content</div>
 <div class="sub-title">
 * Sidebar is rendered with animation effect
 </div>
 <div class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("close_btn").Content("CloseSidebar").Render()
 </div>
</div>).Render();
}
<!-- end of sidebar element -->
<!-- main content declaration -->
<div>
 <div class="title">Sidebar Transitions</div>
 <div class="sub-title"> * Click the below button to render the Sidebar
with animation effect.</div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("zoom").Content("Zoom Sidebar").CssClass("e-
info").Render()
 @Html.EJS().Button("open_door").Content("Open Door").CssClass("e-
info").Render()
 @Html.EJS().Button("bottom_top").Content("Bottom To
Top").CssClass("e-info").Render()
 </div>
 <div style="padding:20px" class="center-align">
 <!-- button element declaration -->
 @Html.EJS().Button("rotate").Content("Rotate").CssClass("e-
info").Render()
 @Html.EJS().Button("rotate_3d").Content("Rotate 3D").CssClass("e-
info").Render()
 @Html.EJS().Button("reverse").Content("Reverse Slide
Out").CssClass("e-info").Render()
 </div>
</div>
<script type="text/javascript">
 document.addEventListener('DOMContentLoaded', function () {
 //create instances for sidebar element
 var sidebarElement = document.getElementById("sidebar-
element").ej2_instances[0];
 sidebarElement.element.classList.add("sidebar");
 // Zoom sidebar
 document.getElementById('zoom').onclick = function () {
 sidebarElement.show();
 sidebarElement.element.classList.add("w3-animate-zoom");
 }
 });

```

```

// Open Door
document.getElementById('open_door').onclick = function () {
 sidebarElement.show();
 var sidebar = document.getElementsByClassName("e-sidebar-
overlay")[0];
 sidebar.classList.add("move");
}
// Bottom to Top
document.getElementById('bottom_top').onclick = function () {
 sidebarElement.show();
 sidebarElement.element.classList.add("w3-animate-bottom");
}
// Rotate sidebar
document.getElementById('rotate').onclick = function () {
 sidebarElement.show();
 sidebarElement.element.classList.add("rotate");
}
// Rotate 3D sidebar
document.getElementById('rotate_3d').onclick = function () {
 sidebarElement.show();
 sidebarElement.element.classList.add("rotate_3d");
}
// Reverse Slide Out
document.getElementById('reverse').onclick = function () {
 sidebarElement.show();
 sidebarElement.element.classList.add("reverse_slide_out");
}
// Close the sidebar
document.getElementById('close_btn').onclick = function () {
 sidebarElement.element.classList.remove("sidebar");
 sidebarElement.element.classList.remove("rotate");
 sidebarElement.element.classList.remove("w3-animate-zoom");
 sidebarElement.element.classList.remove("w3-animate-bottom");
 sidebarElement.element.classList.remove("rotate_3d");
 sidebarElement.element.classList.remove("reverse_slide_out");
 sidebarElement.hide();
}
});
</script>
<style>
/* sample level styles */
.center-align {
 text-align: center;
 padding: 20px;
}
.title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 font-style: italic;
}
.sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 font-style: italic;
}

```



```
.center-align {
 text-align: center;
 padding-top: 20px;
}
/* Animation styles */
.move {
 transform: rotateX(-20deg);
}
.w3-animate-bottom {
 animation-name: animatebottom;
 animation-duration: 1s;
}
@@keyframes animatebottom {
 from {
 margin-top: 100%;
 }
 to {
 margin-top: 0%;
 }
}
.w3-animate-zoom {
 animation-name: animatezoom;
 animation-duration: 1s
}
@@keyframes animatezoom {
 from {
 transform: scale(0)
 }
 to {
 transform: scale(1)
 }
}
.rotate {
 animation-name: rotatel;
 animation-duration: 1s
}
@@keyframes rotatel {
 from {
 transform: rotateX(150deg)
 }
 to {
 transform: rotateX(360deg)
 }
}
.rotate_3d {
 animation-name: rotate;
 animation-duration: 1s
}
@@keyframes rotate {
 from {
 transform: rotateY(150deg)
 }
 to {
 transform: rotateY(360deg)
 }
}
.reverse_slide_out {
```

```

 animation-name: reverse1;
 animation-duration: 1s
 }
 @@keyframes reverse1 {
 from {
 transform: rotateY(-65deg);
 margin-left: 200px;
 }
 to {
 margin-left: 0%;
 }
 }
 /*End of animation styles*/
 /* Button styles */
 #close, #close:hover, #close:active, #close:focus { /* csslint allow:
adjoining-classes*/
 background: #fafafa;
 color: black
 }
 #close_btn, #close_btn:hover, #close_btn:active, #close_btn:focus { /*
csslint allow: adjoining-classes*/
 background: #fafafa;
 color: black
 }
 button {
 margin: 5px;
 }
 /* sidebar element styles */
 .sidebar {
 animation-name: rotate_sidebar;
 animation-duration: 2s
 }
 @@keyframes rotate_sidebar {
 from {
 transform: rotateY(150deg)
 }
 to {
 transform: rotateY(360deg)
 }
 }
 #sidebar-element {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
</style>

```

## ANIMATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller

```

```

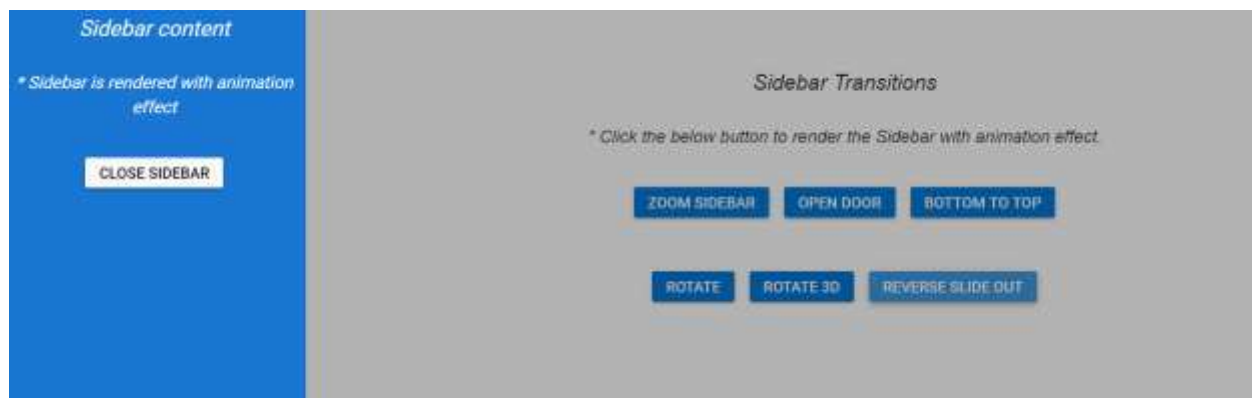
{
 public IActionResult animation()
 {
 return View();
 }
}

```

Output be like the below without animation.



Output be like the below, after applying animation to the Sidebar element.



### Layout Sidebar

The following example demonstrates how to render sidebar in layout page. Sidebar is displayed in all the view page. While navigate to other view page, main content of sidebar changes.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
<!DOCTYPE html>

```

```

<html>
 <head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <title>@ViewBag.Title - My ASP.NET Application</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
 @Styles.Render("~/Content/css")
 @Styles.Render("~/Content/styles.css")
 @Scripts.Render("~/bundles/modernizr")
 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css" />
 <script
src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
 </head>
 <body>
 @{Html.EJS().Sidebar("sidebar-
menu").Width("290px").Target(".maincontent").MediaQuery("(min-
width:670px)").ContentTemplate(@<div>
 <!-- Sidebar content -->
 <div class="sidebar-header header-cover" style="background-
color: #1694CA">
 <div style="text-align:center">
 <div class="sidebar-brand">
 Essential JS 2 Syncfusion Components
 </div>
 </div>
 <div style="padding:0 5px 10px 5px">
 <div class="searchbox">
 <input id="search-by" type="text"
placeholder="Search Components">
 <i class="fa fa-search"
style="font-size:20px"></i>
 </div>
 </div>
 </div>
 <div class="control-section">
 <div class="control_wrapper accordion-control-section">
@Html.EJS().Accordion("accordion").ExpandMode(ExpandMode.Single).EnablePersi
stence(true).Items(new List<AccordionAccordionItem>
{
 new AccordionAccordionItem { Header =
"Navigations",Expanded=true, Content = "#navigation" },
 new AccordionAccordionItem { Header = "Layouts",
Content = "#layouts" },
 new AccordionAccordionItem { Header = "Inputs",
Content = "#inputs" },
}).Render()
 </div>
 <ul id="navigation" style="display:none">
 Sidebar
 ContextMenu
 Accordion


```

```

 <ul id="layouts" style="display:none">
 Avatar
 Card
 Splitter

 <ul id="inputs" style="display:none">
 Input Mask
 TextBox

 </div>
 <!-- end of normal state element declaration -->
</div>).Render();}
<div id="body_content" class="maincontent">
 <div style="padding: 1.5vw 3vw;">
 @RenderBody()
 </div>
</div>
@Html.EJS().ScriptManager()
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>

```

### TEXTBOX.CSHTML

```

@using Syncfusion.EJ2
<table class="table">
 <div id="top-bar">
 <div class="component">
 Components
 <i class="fa fa-angle-right" style="font-size:16px"></i>
 Inputs
 <i class="fa fa-angle-right" style="font-size:16px"></i>
 TextBox
 </div>
 </div>
 <div>
 <div class="content">
 <div class="body-inner">
 TextBox
 </div>
 <div style="text-align:left">
 <div style="font-size:2vw">Overview</div>
 <div class="text">The ASP.NET MVC TextBox (text field)
control is most useful for editing, displaying, or entering plain text on
forms to capture user names, phone numbers, email, and more. This control is
an extended version of the HTML5 TextBox (input type text) control with
icons, floating labels, different sizing, grouping, validation states, and
more. It is available in an HTML5/CSS version and an ASP.NET MVC
version.</div>
 </div>
 <div style="padding:2vw 0 2vw 0;text-align:left">
 <div style="font-size:1.5vw">Key Features</div>
 </div>
 </div>
 </div>
</table>

```

```

 <div class="text">
 <p>1. Floating Label : Floats the placeholder
text while focus.</p>
 <p>2. Input Group : Group the icons, buttons
along with textbox.</p>
 <p>3. Validation States: Provides styles for
success, error, and warning states.</p>
 <p>4. Multiline : Handles multiline input with
placeholder text.</p>
 </div>
 </div>
 <div class="link">
 <div>
 <a
href="https://ej2.syncfusion.com/aspnetmvc/documentation/textbox/getting-
started">Documentation Link
 <a
href="https://ej2.syncfusion.com/aspnetmvc/TextBoxes/DefaultFunctionalities#
/material">Samples Link
 </div>
 </div>
</div>
</table>
<style>
 div a {
 font-size: 1.5vw;
 }
 #top-bar {
 background: #F6F6F6;
 border-radius: 2px;
 height: 0;
 min-height: 40px;
 }
 #body_content {
 margin-top: 0px !important;
 }
 .link {
 height: 50px;
 width: 100%;
 }
 .header {
 font-style: italic;
 font-size: 16px;
 color: #1694CA;
 padding-right: 5px;
 }
 .content {
 text-align: center;
 padding: 2.5vw;
 padding-top: 0px;
 position: relative;
 margin: 2vw;
 box-shadow: 0 0 8px 0;
 }
 .body-inner {
 padding-top: 50px;

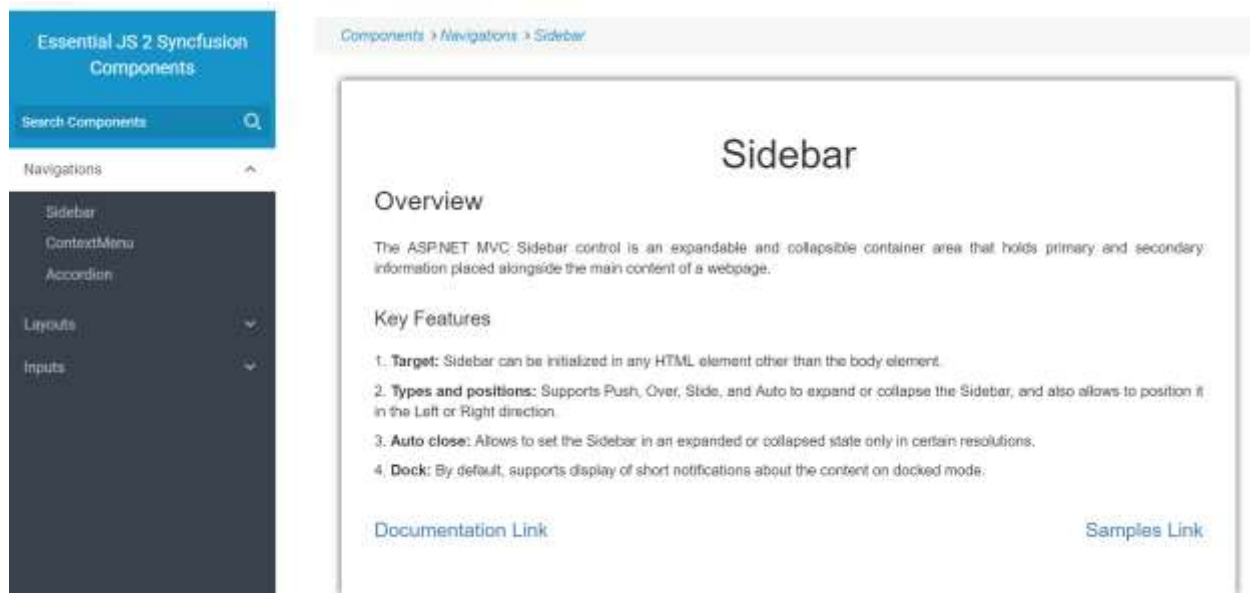
```

```

 text-align: center;
 font-size: 3vw;
 }
 body {
 padding: 0px;
 }
 .component {
 height: 100%;
 line-height: 40px;
 padding: 0 0 0 27px;
 }
 .text {
 font-size: 16px;
 text-align: justify;
 padding-top: 20px;
 font-family: "Muli", "Helvetica", "Tahoma", "Geneva", "Arial", sans-
serif;
 }
</style>

```

Output be like the below.



### Initialize the Sidebar with TreeView

The following example demonstrates how to render TreeView component inside the Sidebar with dock state and how to achieve expand and collapse the functionalities simultaneously in the sidebar and Treeview.

On collapse, the LI elements of TreeView show icons only to represent the short sign of the hidden text content. On expand, hidden text content will be set to be visible.

### CSHTML

```

<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8" />

```

```

<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>@ViewBag.Title - My ASP.NET Application</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
<!-- Syncfusion Essential JS 2 Styles -->
<link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css" />
<!-- Syncfusion Essential JS 2 Scripts -->
<script
src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
</head>
<body>
 <div class="main-header" id="header-section">
 <ul class="header-list">
 <li class="float-left header-style icon-menu" id="hamburger"
(click)="toggle()">
 <li class="float-left header-style nav-pane">Navigation
Pane
 <li class="header-style float-right support border-
left">Support

 </div>
 @{Html.EJS().Sidebar("sidebar-
treeview").Width("290px").Target(".main-
content").EnableDock(true).DockSize("44px").MediaQuery("(min-width:
600px)").Close("onClose").ContentTemplate(@<div>
 <!-- normal state element declaration -->
 <div class="main-menu">
 <div>
 <!-- Treeview element declaration-->
 @Html.EJS().TreeView("main-treeview").Fields(field=>
field.Id("nodeId").Text("nodeText").HasChildren("hasChild").IconCss("iconCss
")
.DataSource(ViewBag.dataSource)).Render()
 </div>
 </div>
 <!-- end of normal state element declaration -->
</div>).Render();}
 <div id="body_content" class="maincontent">
 <div>
 @RenderBody()
 </div>
 </div>
<script>
 document.addEventListener('DOMContentLoaded', function () {
 var sidebarTreeview = document.getElementById("sidebar-
treeview").ej2_instances[0];
 var treeviewObj = document.getElementById("main-
treeview").ej2_instances[0];
 // Toggle the Sidebar
 document.getElementById('hamburger').onclick = function () {
 if (sidebarTreeview.isOpen) {
 sidebarTreeview.hide();
 treeviewObj.collapseAll();
 }
 }
 });

```



```

 else {
 sidebarTreeview.show();
 treeviewObj.expandAll();
 }
 });
 function onClose() {
 var treeviewObj = document.getElementById("main-
treeview").ej2_instances[0];
 treeviewObj.collapseAll();
 }
</script>
@Html.EJS().ScriptManager()
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
<style>
/* header-section styles */
#header-section.main-header {
 border-bottom: 1px solid #d2d6de;
 height: 55px;
 min-height: 55px;
 max-height: 55px;
 background: #1c86c8;
 color: #fff;
}
#header-section .header-style {
 line-height: 40px;
 height: 55px;
 padding: 8px;
 list-style: none;
 text-align: center;
 font-size: 18px;
}
#header-section .border-left {
 border-left: 1px solid #d2d6de;
 width: 10em;
}
#header-section .float-left {
 float: left;
 padding-right: 0px;
}
#header-section .icon-menu {
 width: 40px;
 cursor: pointer;
}
/*end of header-section styles */
#sidebar-treeview {
 border-left: 0.5px solid #80808070;
 border-bottom: 0.5px solid #80808070;
}
/*main-menu-header styles */
#sidebar-treeview .main-menu .main-menu-header {
 color: #656a70;
 padding: 15px;
 font-size: 14px;
 width: 13em;
}

```

```

 margin: 0;
 }
 /*end of main-menu-header styles */
 /*text input styles */
 #sidebar-treeview .main-menu .search-icon {
 text-indent: 10px;
 height: 30px;
 width: 19em;
 }
 /*end of text input styles */
 /* table of content area styles */
 #sidebar-treeview .table-content {
 padding: 20px;
 height: 8em;
 }
}
/* end of table ofcontent area styles */
/* content area styles */
#main-text.main-content {
 overflow: hidden;
}
#main-text .sidebar-content .line {
 width: 100%;
 height: 1px;
 border-bottom: 1px dashed #ddd;
 margin: 40px 0;
}
#main-text .sidebar-content {
 padding: 15px;
}
#main-text .sidebar-heading {
 color: #1c86c8;
 margin: 40px 0;
 padding: 2px;
}
#main-text .paragraph-content {
 font-family: 'Poppins', sans-serif;
 padding: 5px;
 font-weight: 300;
 color: grey;
}
/* end of content area styles */
/* end of body and html styles */
/* icon styles */
@@font-face {
 font-family: 'fontello';
 src: url('data:application/octet-
stream;base64,AAEAAAAPAIAAAwBwRlNVQiCLJXoAAAD8AAAAVE9TLzI+JUkyAAABUAAAFZjbW
Fw0almQAAAAagAAAIgY3Z0IAbV/vwAABfUAAAAIGZwZ22KkZBZAAAX9AAAC3BnYXNwAAAAEAAAF8
wAAAAIZ2x5Zk30JrMAAAPIAAAPrGhlyWQTW6AfAAATdAAAADZoaGVhB2gDnAAAE6wAAAAkaG10eD
Hm//YAABPQAAAAOGxvY2EejhqYAAAUCAAAAB5tYXhwAfYMkAAAFcGAAAAGbmFtZcydHiAAABRIAA
ACzXBvc3RuKDzPAAAXGAAALRwcmVw5UErVAAAI2QAAACGAAEAAAAKADAAPgACREZMVAAObGF0bg
AaaaQAAAAAAAAAAQAAAAQAAAAAAAAAAQAAAFsaWdhAAgAAAABAAAAAQAEAAQAAAAABAaGAAQAGAA
AAAQAAAAEDkAGQAAUAAAJ6ArwAAACMAAoCvAAAAeAAMQECAAACAAUDAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAFBmRWQAQOgB6BMDUv9qAFoDUgCaAAAAAQAAAAAAAAAAUAAAADAAAAALAAAAQAAAF0AA
EAAAAAG4AAwABAAAAALAADAAoAAAF0AAQAQgAAAAAYABAABAAALoCegT//8AAOgB6BD//wAAAAAAQ
AGABYAAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAANAAABBgAAAAAAAAAAAAAAAAAAAAAAAAAA
AAA

```

2823

5qkKKObjoEQmaWTXtkvxUAAAAAAv/9/2oDWQNSACYATQA8QDlFQj8NBwUGAAFLSEY+DgUDACIaAg  
IDA0cAAAEDAQAdBQABAQxIAAMDAlgaAGINakksKyAeFxIEBRYrET4BNzYXNjc1PgEyFhcTNhceAQ  
cOAgcOAgcVFAYHISImJzU0LgE3HgIXITU+ATc+AT8BMjY3NicuAQ4BBxEuAScOAgcVJgcmBgcmBg  
JKSTNEGSACRmtEBQFeTdc2FxdwFRciUhemGf6lGiQDhBY+AhYcAQFbEG4NFUIWRQQAQNFkg8WB  
YCIhwYIgMxOhpCDj46AaM8TAQRChAGazVMSDn+7y0cE3Y4FhALDipMFpsZJAMMGqochHQDn2x6Fw  
MmYhMZIAQNAgQVGiMOFiIDAW0bJAICJBu/MTsQEhsJOAAAAGAA/74CygMLAAUAIgAyQC8UBQMCBA  
IAAUcDAQIAAnAEAEQAAAAFUBAEBAQBWAAABAEoHBhgWEhAGIGcheAUFFSsBIREBHwETMhceARcRFA  
YHBiMiLWEHBiMiJy4BNRE0Njc2MwKD/cQBHjLsBwwMEXQBFhIKDhsU9vYUGg0MEHYWEGwNAsP9Sw  
ESL+MC/QUIHhT9MRMGbWQS7OwTBQcgEwLPEyAHBQAABgAA/2oDWQNSABMAGgAjADMAQWbTAHJAbx  
QBAGQsJAIHBkA4AggJUEgCCgsERwACAAMGAgNgAAYABwkGB2ANAQkACAsJCGAOAQsACgULCmAABA  
QBWAABAQxIDAEBQBYAAAAQBJREQ0NBsbRFNEUkxKNEM0Qjw6MC4oJhsjGyMTJhQ1Ng8FGSsBHg  
EVERGGBYeiJicRNDY3ITIWFwcVMYyVASYTESMiJic1IRETNDYzITIWHQEUBiMhIiY1BTIWHQEUBi  
MhIiY9ATQ2MwUyFh0BFAYjISImPQE0NjMDMxAWHhf9EhceASAWafQWNg9K0gUHRwbG6BceAf5Tjw  
oIAYkICgoI/ncICgGbCAoKCP53CAoKcAGJCAoKCP53CAoKCAJ+EDQY/X4XHgEgFgN8Fx4BFhAm0h  
EGrwf8sAI8IBXp/KYB4wcKCgkCAoKCFkKCCQICgoIJAgKjwoIJAgKCgkCAoAAAAA//9/7EDXw  
MLAA8ANWBEAEhARskBBQMJAQIBAAJHAAQCAwIEA20AAwUCAwVrAAcAAgQHAmAABQAAAQUAYAABBg  
YBVAABAQZYAAAYBBkwVHisTFiYmIwgFHCslNTQmKWeiBh0BFBY7ATI2EzQuASMiBWyfARYzMjc+AT  
IWRQGBW4BFxUUFjsBMjY0Nj8BPgMXFA4Bii4CPgEyHgEB9AoIawgKCghrCAqPPlwxiEcJDUoEBg  
kFHiU4KhYbIzWBCghrCAoYehwKHhQM13LG6MhuBnq89Lp+UmsICgoIawgKCgF/MVQudw0LNwQHJh  
seEhUaDA9CJRQICgoSigsQBhocKFJ1xHR0xOrEdHTEAAEAAAABAACCKpnPXw889QALA+gAAAAA2E  
iuQQAADYDK5B//3/ZgQWA1IAAAIAIAIAAAAAAAAAAQAAL/agAABC///f/0BBYAAQAAAAAAAA  
AAAAAAAAAAAA4D6AAAA1kAAAPoAAAD6AAABC8AAAOgAAADWQAAA+gAAANZ//0DoP//A03//QLKAA  
ADWQAAA1n//QAAAAAAZgD6AegCWgMABegFHAVkBBIGSAacB1AH1gAAAAEAAAAOALAACwAAAAAAG  
BeAG4AcwAAQsLcAAAAAAAAASAN4AAQAAAAAAAAA1AAAAQAAAAAAAAQAIADUAAQAAAAAAAAAGAHAD  
0AAQAAAAAAAAAwIAEQAAQAAAAAAAAABAAIAEwAAQAAAAAAAAABQALAFQAAQAAAAAAAAABGAIaf8AAQAAAAAACg  
ArAGcAAQAAAAAACwATAJIAAwABBAkAAABqAKUAAwABBAkAAQAQAQ8AAwABBAkAAAGAOAR8AAwABBA  
kAAwAQAS0AAwABBAkABAAQAT0AAwABBAkABQAWAU0AAwABBAkABgAQAWMAAwABBAkACgBWAXMAAw  
ABBAkACwAmAc1Db3B5cmlnaHQgKEMpIDIwMTggYnkqb3JpZ2luYWwgYXV0aG9ycyBAIGZvbnRlbG  
xvLmNvbWZvbnRlbGxvUmVndWxhcmZvbnRlbGxvZm9udGVsbG9WZXJzaW9uIEdEuMGZvbnRlbGxvR2  
VuZXJhdGvKIGJ5IHN2ZzJ0dGYgZnJvbSBGbz250ZWxsbyBwcm9qZWN0Lmh0dHA6Ly9mb250ZWxsby  
5jb20AQwBvAHAAeQBYAgkAZwBoAHQAIAAoAEMAQKAgADIAMAAXADgAIABiAHkAIABvAHIAaQBNAG  
kAbgBhAGwAIAbAHUADABoAG8AcgBzACAAQAAGAGYAbwBuAHQAZQBzAGwAbwAuAGMabwBtAGYAbw  
BuAHQAZQBzAGwAbwBSAGUAZwB1AGwAYQBYAGYAbwBuAHQAZQBzAGwAbwBmAG8AbgB0AGUAbABsAG  
8AVgB1AHIAcWBPAG8AbgAgADEALgAwAGYAbwBuAHQAZQBzAGwAbwBHAGUAbgB1AHIAIYQB0AGUAZA  
AgAGIAeQAgAHMAdgBnADIAdAB0AGYAIABmAHIAbwBtACAARgBvAG4AdABLAGwAbABvACA4AYwBvAG0AAAAAGAAAA  
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAOAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDw  
AEbWVudQRkb2NzAnRoBGNvZGUNdGh1bWJzLXVwLWFsdAltaWNyb2NoaXAGY2hyb211c2NpcmNsZS  
10aGluCHN1YXJjaC0xB3VwLWVhbmQOYm9va21hcmstZW1wdHkiZG9jLXRleHQMaGVscC1jaXJjbG  
VkAAAAAQAB//8ADwAAAAAAAAAAAAAAAAAAAAAAAAAGAAAYABgAGANS/2YDUv9msAAsILAAVvHfWSAgS7  
gADlFLsAZTWliwNBuwKFlgZiCKVViwAiVhuQgACABjYyNiGyEhsABZsABDI0SyAAEAQ2BCLbABLL  
AgYGYtsAIsIGQgsMBQsAQmWrIoAQpDRWNFULtYISMhG4pYILBQUFghsEBZGyCwOFBYIbA4WVksQ  
EKQ0VjRWFksChQWCGxAgpDRWNFILAwUFghsDBZGyCwWFBYIGYgiophILAKUFhgYcWIFBYIbAKYB  
sgsDZQWCGwNmAbYFlZWruWastZWSowAFBYZVlZLbADLCBFILAEJWFkILAFQ1BYsAUjQRAGI0IbIS  
FZsAFgLBaELCMhIyEgZLEFYkIgsAYjQREBCKNFY7EBCKowAWBFY7ADKiEgsAZDIIogirABK7EwBS  
WwBCZRWBQ2FQSWvgjWSEgSEBTWLABKxshsEBZI7AAUFhlWS2wBSyWb0MrsgACAENGQi2wBiywBy  
NCIyCwACNCyBACyMAwAWowAWCwBSotsAcSICBFILALQ204BABiILAAUFIWQBZzrABY2BESAFgLB  
AIIILHCwBDRUIqIbIAAQBDYEItsAkssABDI0SyAAEAQ2BCLbAKLCAGrSCwASsjsABDsAsQj1YCBFii  
NhIGQgsCBQWCGwABwMFBYsCabsEBZWSowAFBYZVmwAyUjYUREsAFgLBaLLCAGrSCwASsjsABDsA  
Q1YCBFiiNhIGSwJFBYsAABsEBZI7AAUFhlWbADJSNhrESwAwatsAwsILAAI0KyCwoDRVghGyMhWS  
ohLbANLLECAkKwZGFELbAOLLABYCAgsAxDSrAAUFggsAwjQlmwDUNKsABSWCCwDSNCWS2wDyWgsB  
BiZrABYyC4BABjiiNhsA5DYCKYCCwDiNCIy2wECxLVFixBGREWSSwDWUjeC2wESxLUVhLU1ixBG  
REWRshWSSwE2UjeC2wEiyxAA9DVVixDw9DsAFhQrAPK1mwAEOWAiVCsQwCJUKxDQILqRABFiMgsA  
MLUFixAQBDYLAEJUKKiickI2GwDiohI7ABYSCKI2GwDiohG7EBAENGsAILqRACJWGWdDiohWbAMQ0  
ewDUNHYLACYiCwAFBYsEBgWWawAWMgsAtDY7gEAGIGsABQWLBAYFlmsAFjYLEAABmjRLABQ7AAPr  
IBAQFDYEItsBMsALEAAKVUWLAPI0IgRbALI0KwCiOwAWBCIGCwAWG1EBABAA4AQkKKYLESBiuwci  
sbIlktsBQssQATKy2wFSyxARMrLbAWLLECEytsBcssQMTKy2wGCyxBBMrLbAZLLEFEytsBossQ

```

YTKy2wGyyxBxMrLbAcLLEIEystsB0ssQkTKy2wHiwAsA0rsQACRVRYSa8jQiBFsAsjQrAKI7ABYE
IgYLABYbUQEAEADgBCQopgsRIGK7ByKxsiWS2wHyxxAB4rLbAgLLEBHistsCEssQIEKy2wIiyxAx
4rLbAjLLEEHistsCQssQUeKy2wJSyxBh4rLbAmLLEHHistsCcscQgeKy2wKCyxCR4rLbApLCA8sA
FgLbAqLCBgSBBgIEMjsAFgQ7ACJWGwAWCkSohLbArLLAqK7AqKi2wLCwgIEcgILALQ2O4BABiIL
AAUFIwQGBZzrABY2AjYTgjIIPVWCBHICcWc0NjuAQAYiCwAFBYsEBgWWawAWNgI2E4GyFZLbAtLA
CxAAJFVFIwARawLCqWARUwGyJZLbAuLACwDSuxAAJFVFIwARawLCqWARUwGyJZLbAvLCA1sAFgLb
AwLACwAUvjuAQAYiCwAFBYsEBgWWawAWOWASuWC0NjuAQAYiCwAFBYsEBgWWawAWOWASuWABa0AA
AAAABEPiM4sS8BFSotsDEsIDwgRyCwC0NjuAQAYiCwAFBYsEBgWWawAWNgSABDYTgtsDIsLhc8Lb
AzLCA8IEcgsAtDY7gEAGIgsABQWLBAYFlmsAFjYLAAQ2GwAUNjOC2wNCyxAgAWJSAuIEewACNCsA
ILSYqKRYNHI2EgWGIBIVmwASNCsjMBARUUKi2wNSywABawBCWwBCVHI0cjYbAJQytlii4jICA8ij
gtsDYssAAWsAqlsAqlIC5HI0cjYSCwBCNCsAlDKyCwYFBIYILBAUVizAiADIBuzAiYDGLlCQiMgsA
hDIIojRyNHI2EjRmCwBEOwAmIgsABQWLBAYFlmsAFjYCCwASsgioPhILACQ2Bki7ADQ2FkUFIwAk
NhG7ADQ2BZsAmlsAJiILAAUFIwQGBZzrABY2EjICCwBCYjRmE4GyOwCENGsAilSahDRyNHI2FgIL
AEQ7ACYiCwAFBYsEBgWWawAWNgIyCwAssjsARDYLABK7AFJWGwBSWwAmIgsABQWLBAYFlmsAFjsA
QmYSCwBCVgZCOWAyVgZFBYIRsjIVkjICCwBCYjRmE4WS2wNyywABYgICCwBSYgLkCjRyNhIzw4Lb
A4LLAAFIcWCCNCICAgRiNHsAErI2E4LbA5LLAAFrADJbACJUcjRyNhsABUWC4gPCMhG7ACJbACJU
cjRyNhILAFJbAEJUcjRyNhsAYlsAUlSbACJWG5CAAIAGNjIyBYyHshWWO4BABiILAAUFIwQGBZzr
ABY2AjLiMgIDyKOCMhWS2wOiywABYgsAhDIC5HI0cjYSBgsCBgZrACYiCwAFBYsEBgWWawAWMjIC
A8ijgtsDssIyAuRrACJUZWCA8WS6xKwEUKy2wPCWjIC5GsAilRlBYIDxZLrErARQrLbA9LCMgLk
awAiVGULggPFkjcIC5GsAilRlBYIDxZLrErARQrLbA+LLA1KyMgLkAwAiVGULggPFkusSsBFCstsD
8ssDYriiAgPLAEI0KKOCMgLkAwAiVGULggPFkusSsBFCuwbEMusCsrLbBALLAAFrAEJbAEJiAuRy
NHI2GwCUMrIyA8IC4jOLErARQrLbBBLLEIBCVCSAAWsAqlsAqlIC5HI0cjYSCwBCNCsAlDKyCwYF
BYILBAUVizAiADIBuzAiYDGLlCQiMgR7AEQ7ACYiCwAFBYsEBgWWawAWNgILABKyCKimEgsAJDYG
QjsANDYWRQWLACQ2EbsANDYFmwAyWwAmIgsABQWLBAYFlmsAFjYbACJUZhOCMgPCM4GyEgIEYjR7
ABKyNhOCFZsSsBFCstsEIssDUrLrErARQrLbBDLLA2KyEjICA8sAQjQiM4sSsBFCuwbEMusCsrLb
BELLAAFSBHsAAjQrIAAQEVFBMusDEqLbBFLLAAFSBHsAAjQrIAAQEVFBMusDEqLbBGLLEAARQTsD
IqLbBHLLA0Ki2wSCyWABZFIyAuIEaKI2E4sSsBFCstsEkssAgjQrBIKy2wSiyyAABBKy2wSyyyAA
FBKy2wTCyyAQBBKy2wTSyyAQFBKy2wTiyyAABCKy2wTyyyAAFCKy2wUCyyAQBCKy2wUSyyAQFCKy
2wUiyyAAA+Ky2wUyyyAAE+Ky2wVCyyAQA+Ky2wVSyyAQE+Ky2wViyyAABAKy2wVyyyAAFAKy2wWC
yyQBAKy2wWSyyAQFAKy2wWiyyAABDKy2wWyyyAAFDKy2wXCyyAQBDKy2wXSyyAQFDKy2wXiyyAA
A/Ky2wXyyyAAE/Ky2wYCYyAQA/Ky2wYSyyAQE/Ky2wYiywNysusSsBFCstsGMssDcrsDsrLbBkLL
A3K7A8Ky2wZSywABawNyuwPSstsGYssDgrLrErARQrLbBnLLA4K7A7Ky2waCywOCuWPCstsGkssD
grsD0rLbBqLLA5Ky6xKwEUKy2wayywOSuW0ystsGwssDkrsDwrLbBtLLA5K7A9Ky2wbiywOisusS
sBFCstsG8ssDorsDsrLbBwLLA6K7A8Ky2wcSywOiuwPSstsHIsswKEAgNFWCEbIyFZQiuwCGWwAy
RQeLABFTAtAEu4AMhSWLEBAY5ZsAG5CAAIAGNwsQAFQrIAAQAgSQAfQrMKAgEIKrEABUKzDgABCC
qxAAZCugLAAAEACsQxAAdCugBAAAEACsQxAwBESQBIFFYsECIWLEDZESxJgGIUVi6CIAAAQRAiG
NUWLEDAERZwVlZswWCAQwquAH/hbAEjBECAEQAAA==') format('truetype');
}
#sidebar-treeview #main-treeview .icon {
 font-family: 'fontello';
 font-size: 16px;
 margin-top: -4px;
}
#header-section.main-header .icon-menu::before {
 content: '\e801';
 font-family: 'fontello';
 font-size: 27px;
}
#sidebar-treeview #main-treeview .icon-microchip::before {
 content: '\e806';
}
#sidebar-treeview #main-treeview .icon-thumbs-up-alt::before {
 content: '\e805';
}
#sidebar-treeview #main-treeview .icon-docs::before {
 content: '\e802';
}
}

```

```

#sidebar-treeview #main-treeview .icon-th::before {
 content: '\e803';
}
#sidebar-treeview #main-treeview .icon-code::before {
 content: '\e804';
}
#sidebar-treeview #main-treeview .icon-chrome::before {
 content: '\e807';
}
#sidebar-treeview #main-treeview .icon-up-hand::before {
 content: '\e810';
}
#sidebar-treeview #main-treeview .icon-bookmark-empty::before {
 content: '\e811';
}
#sidebar-treeview #main-treeview .icon-help-circled::before {
 content: '\e813';
}
#sidebar-treeview #main-treeview .icon-doc-text::before {
 content: '\e812';
}
#sidebar-treeview #main-treeview .icon-circle-thin::before {
 content: '\e808';
}
#header-section .header-list, #sidebar-treeview .e-treeview,
#sidebar-treeview .e-treeview .e-ul {
 padding: 0;
 margin: 0;
 overflow: hidden;
}
.e-content-animation {
 border-right: 0.5px solid #80808070;
 border-bottom: 0.5px solid #80808070;
}
body {
 padding: 0px;
}
.e-sidebar.e-open #main-treeview .e-text-content {
 padding: 0 0 0 24px;
}
.e-sidebar.e-close #main-treeview .e-text-content {
 padding: 0 0 0 8.5px;
}
#main-treeview ul {
 overflow: inherit;
}
/* end of icon styles */
</style>
</body>
</html>

```

## INDEX.CSHTML

```

<div class="main-content" id="main-text">
 <div class="sidebar-content">
 <h2 class="sidebar-heading"> Responsive Sidebar With Treeview</h2>

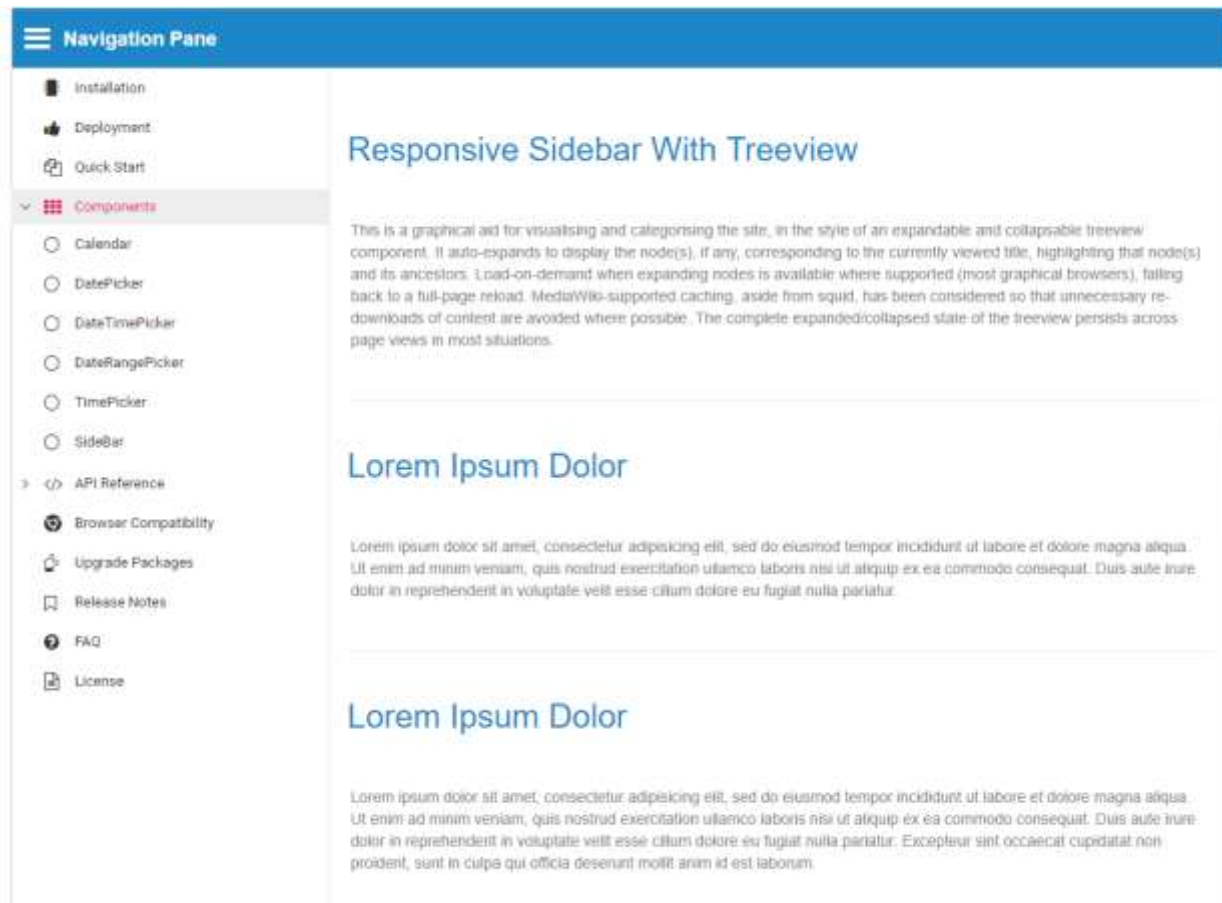
```

```

 <p class="paragraph-content">
 This is a graphical aid for visualising and categorising the
 site, in the style of an expandable and
 collapsable treeview component. It auto-expands to display the
 node(s), if any, corresponding to the currently
 viewed title, highlighting that node(s) and its ancestors. Load-
 on-demand when expanding nodes is available
 where supported (most graphical browsers), falling back to a
 full-page reload. MediaWiki-supported caching,
 aside from squid, has been considered so that unnecessary re-
 downloads of content are avoided where possible.
 The complete expanded/collapsed state of the treeview persists
 across page views in most situations.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading">Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
 enim ad minim veniam, quis
 nostrud exercitation ullamco laboris nisi ut aliquip ex ea
 commodo consequat. Duis aute irure
 dolor in reprehenderit in voluptate velit esse cillum dolore eu
 fugiat nulla pariatur.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod
 tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
 minim veniam, quis nostrud
 exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat. Duis aute irure dolor in
 reprehenderit in voluptate velit esse cillum dolore eu fugiat
 nulla pariatur. Excepteur sint
 occaecat cupidatat non proident, sunt in culpa qui officia
 deserunt mollit anim id est laborum.
 </p>
 </div>
</div>

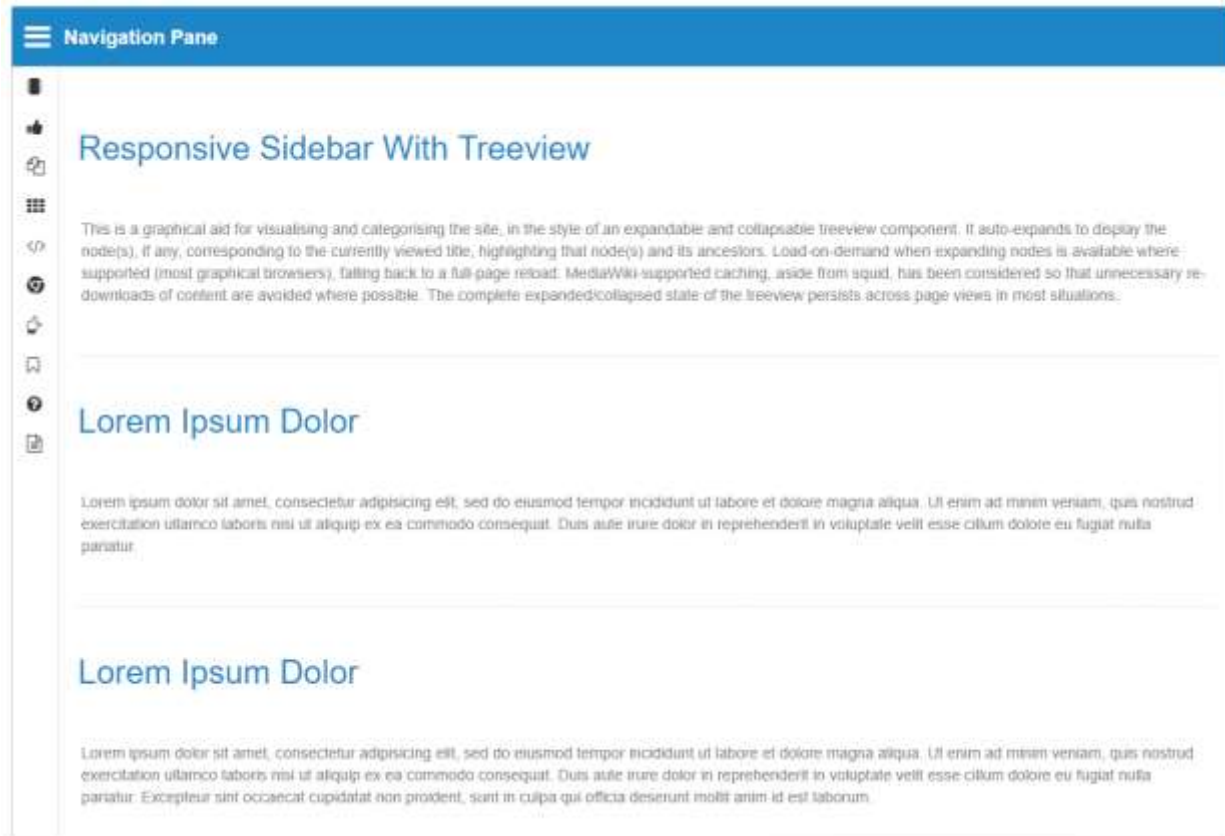
```

Output be like the below in Expanded state.



Output be like the below in Collapsed state.





### Layout Sidebar using Content Template

In the following example, Menu component is rendered inside the Sidebar using content template. Initially, the Sidebar renders in the dock state with icons, and expands when the hamburger icon at the top-left corner of the header section is clicked.

### CSHTML

```
@using Syncfusion.EJ2
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>@ViewBag.Title - My ASP.NET Application</title>
 <!-- Syncfusion Essential JS 2 Styles -->
 <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
 <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet" />
 <link href="https://cdnjs.cloudflare.com/ajax/libs/highlight.js/9.12.0/styles/default.min.css" rel="stylesheet" />
 <link href="https://cdn.syncfusion.com/ej2/material.css" rel="stylesheet" />
 <!-- Syncfusion Essential JS 2 Scripts -->
```

```

<script
src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
</head>
<body>
 <div class="header-section dock-menu" id="header">
 <ul class="header-list">
 <li id="hamburger" class="icon-menu icon list"
(click)="toggle()">
 <input type="text" placeholder="Search..." class="search-
icon list">
 <li class="right-header list">
 <div class="horizontal-menu">
 <!-- menu element declaration-->
 @Html.EJS().Menu("horizontal-
menubar").Items(ViewBag.AccountMenuItems).CssClass("dock-menu").Render()
 </div>

 <li class="right-header list support">Support
 <li class="right-header list tour">Tour

 </div>
 <!-- sidebar element declaration-->
 @{Html.EJS().Sidebar("sidebar-menu").Width("220px").Target(".main-
content").EnableDock(true).DockSize("50px").ContentTemplate(@<div
class="dock-menu">
 <!-- normal state element declaration -->
 <div class="main-menu">
 <p class="main-menu-header">MAIN</p>
 <div>
 <!-- menu element declaration-->
 @Html.EJS().Menu("main-
menubar").Items(ViewBag.mainMenuItems).Orientation(Syncfusion.EJ2.Navigation
s.Orientation.Vertical).CssClass("dock-menu").Render()
 </div>
 </div>
 <div class="action">
 <p class="main-menu-header">ACTION</p>
 <button class="e-btn action-btn" id="action-button">+
Button</button>
 </div>
 <!-- end of normal state element declaration -->
</div>).Render();}
 <div id="maintext" class="main-content">
 <div>
 @RenderBody()
 </div>
 </div>
 <script>
 document.addEventListener('DOMContentLoaded', function () {
 sidebarInstance = document.getElementById("sidebar-
menu").ej2_instances[0];
 // Expand the Sidebar

 document.getElementById('hamburger').addEventListener('click', function () {
 sidebarInstance.toggle();
 });
 });
 </script>

```

```

 });
 });
</script>
@Html.EJS().ScriptManager()
@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
<style>
 /* header-section styles */
 #header.header-section,
 #header .search-icon {
 height: 50px;
 }
 #hamburger.icon-menu {
 font-size: 24px;
 float: left;
 line-height: 50px;
 padding-top: 8px;
 }
 #header .right-header {
 height: 35px;
 padding: 7px;
 float: right;
 }
 #header .list {
 list-style: none;
 cursor: pointer;
 font-size: 16px;
 line-height: 35px;
 }
 #header .header-list {
 padding-left: 15px;
 margin: 0;
 }
 @@media(max-width:500px) {
 #header .right-header.list.support,
 #header .right-header.list.tour {
 display: none;
 }
 }
 /* text input styles */
 #header .search-icon {
 float: left;
 padding-left: 15px;
 border: 0px solid #33383e !important;
 background-color: #33383e;
 cursor: text;
 width: 5em;
 }
 #header .search-icon:focus {
 outline: none;
 cursor: default;
 }
 /* end of text input styles */
 /* end of header-section styles */
 /* content area styles */
 #maintext.main-content {

```

```

 height: 100vh;
 z-index: 1000;
 }
 #maintext .content {
 margin-top: 230px;
 text-align: center;
 font-size: 32px;
 color: #1784c7;
 }
 /* end of content area styles */
 /* menu styles */
 /* horizontal-menu styles */
 #header .header-list .horizontal-menu .e-menu-item {
 height: 35px;
 vertical-align: middle;
 font-size: 16px;
 line-height: 35px;
 }
 #header .e-menu-item .e-caret {
 line-height: 35px;
 }
 /* end of horizontal-menu styles */
 /* vertical-menu styles */
 #sidebar-menu .e-menu-wrapper ul .e-menu-item.e-menu-caret-icon
{
 width: 220px;
 }
 #sidebar-menu .e-menu-wrapper ul .e-menu-item:hover, .e-menu-
item.e-focused:hover {
 background-color: #3e454c !important;
 }
 /* end of vertical-menu styles */
 /* end of menu styles */
 /* Sidebar styles */
 /* docksidebar styles */
 .dock-menu .e-menu-wrapper ul .e-menu-item .e-caret,
 #header .search-icon,
 #sidebar-menu .action-btn,
 #header .e-menu-item .e-caret,
 .dock-menu .e-menu-wrapper ul .e-menu-item {
 color: #fff !important;
 }
 .dock-menu.e-close .e-menu-wrapper ul .e-menu-item {
 width: 50px;
 }
 .dock-menu.e-close ul .e-menu-item.e-menu-caret-icon {
 padding-right: 12px;
 }
 #sidebar-menu.e-dock.e-close .e-menu-wrapper ul .e-menu-item .e-
caret,
 #sidebar-menu.e-dock.e-close .main-menu-header,
 #sidebar-menu.e-dock.e-close .action-btn {
 display: none;
 }
 #sidebar-menu.e-dock.e-close .e-menu-wrapper ul .e-menu-item.e-
menu-caret-icon,
 #sidebar-menu.e-dock.e-close .e-menu-wrapper ul.e-vertical {

```

```

 min-width: 0;
 width: 50px !important;
 }
 #sidebar-menu.e-dock.e-close .e-menu-wrapper ul.e-menu {
 font-size: 0;
 }
 #sidebar-menu.e-dock.e-close .e-menu-item .e-menu-icon {
 font-size: 20px;
 padding: 0;
 }
 #sidebar-menu,
 #sidebar-menu ul,
 #header ul,
 .dock-menu .e-menu-wrapper,
 .dock-menu.e-menu-wrapper,
 .dock-menu.e-menu-wrapper ul > *,
 .dock-menu .e-menu-wrapper ul > * {
 background-color: #33383e !important;
 color: #fff !important;
 overflow: hidden;
 }

 /* end of docksidebar styles */
 /*end of Sidebar styles */
 /*main-menu-header styles */
 #sidebar-menu .main-menu-header {
 padding: 4px 0px 0px 18px;
 color: #656a70;
 }
 /*end of main-menu-header styles */
 /*button styles */
 #sidebar-menu .action-btn {
 margin-left: 16px;
 width: 165px;
 height: 30px;
 font-size: 13px;
 border-radius: 5px;
 }
 #sidebar-menu .action-btn {
 background-color: #1784c7;
 }

 /* custom code start */
 /*end of button styles */
 .col-md-12, body {
 padding: 0;
 }
 #sidebar-menu {
 margin-left: -1px;
 }
 /*body styles */
 /* custom code end */
 /*end of body styles */
 /*icon styles */
 @@font-face {
 font-family: 'fontello';
 src: url('data:application/octet-
stream;base64,AAEAAAAPAIAAAwBwRlNVQiCLJXoAAAD8AAAAVE9TLzI+Ik1CAAABUAAAAFZjbW
FwkivVUAAAAagAAAI5Y3Z0IAbX/wIAABFMAAAAIGZwZ22KkZBZAAARbAAAC3BnYXNwAAAAEAAEU

```

QAAAAIZ2x5ZmjN+4gAAAO8AAAJRGlYwQUVp+lAAANAAAAADZoaGVhB+UEBwAADTgAAAAkaG10eC  
8e//EAAA1cAAAANGxvY2EOPhBsAAANkAAAABxtYXhwAPsL9gAADawAAAAGbmFtZcydHyEAAA3MAA  
ACzXBvc3ReFbn+AAQnAAAAKVwcmVw5UErvAAAHNwAAACGAEEAAAkADAApGACREZMVAaObGF0bg  
AaAAQAAAAAAAAAAAAQAAAAQAAAAAAAAAAAAFsaWdhAAgAAAABAAAAQAEEAAQAAAAABAAGAAQAGAA  
AAAQAAAAEDoAGQAAUAAAJ6ArwAAACMAncvAAAAeAAMQECAAACAAUDAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAFBmRWQAQOGb6BIDUv9qAFoDUwCXXXXAAQAAAAAAAAAAAAUAAADAAAAALAAAAQAAAFyAA  
EAAAAAGwAAwABAAAAALADAAoAAAFyAAQAQAAAAAYABAABAAALoCegS//8AAOGb6BD//wAAAAAAQ  
AGABYAAAAABAAIAAwAEAAUABgAHAAGACQAKAAsADAAAAQYAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA  
AAAAAAAAAAAAADAAAAAaAAAAAAAMAAADoAQAA6AEAAAABAAADoAgAA6AIAAAACAADoAwAA6AMAAA  
ADAADoBAAA6AQAAAAEAAADoBQAA6AUAAAAFAADoBgAA6AYAAAAGAADoBwAA6AcAAAAHAADoCAAA6A  
gAAAAIAADoCQAA6AkAAAAJAADoEAAA6BAAAAKAADoEQAA6BEAAAAAALADoEgAA6BIAAAAMAAAAAv  
/9/2oDWQNSACYATQA8QDlFQj8NBwUGAAFLSEY+DgUDACIAaAgIDA0cAAEDAQADbQABAQxIAAMDAl  
gAAGINakksKyAeFxiEBRYrET4BNzYXNjc1PgEyFhcTNhceAQcOAQcOAgcVFAYHISImJzU0LgE3Hg  
IXITU+ATc+AT8BMjY3NicuAQ4BBxEuAScOAQcVJgcmBgcmBgJKSTNEGSACRmtEBQFeTDC2FxdwFR  
ciUhEmGf6lGiQDhBY+AhYcAQFbEG4NFUIWRQQAQGNfkg8WBYCIhwYIgMxOhpCDj46AaM8TAQRCh  
AGazVMSDn+7y0cE3Y4FhALDipMFpsZJAMMgqochHQdn2x6FwMmYhMZIAQNAgQVGiMOFiIDAW0bJA  
ICJBu/MTsQEhsJOAAAAGAA/2oDxANTAAwANAA/QDwaDQIBBgABAgACRwABBgMGAQNTBQEDAAYDAG  
sAAAIgAAJrAAYGDEgAAgIEWAAEBA0ESR8iEiMjExIHBRsrBTQjIiY3NCIVFbY3MiUUBisBFAYiJj  
UjIiY1PgQ3NDY3JjU0PgEWFQRHHGEXFB4DAF0JITABEjooCQHhKh36VHZU+h0qHC4wJBICgKfIC  
wgBWqCARYiMDBGCDaHcQkpOgGpHS07VFQ7Kh0YMLReiE1UkhAKCxcAiIVCwoQklR0hmBSNAACAA  
D/sQLKAwwAFQAeACVAIGAFaQVvAwEBBAFvAAQCBG8AAgACbwAAAGYTFxERFzIGBRorJRQGIYeiJj  
U0PgMXFjI3Mh4DaxQGIi4BNh4BAspGMf4kMUYKGC0+LUnKSipCJhwIj3y0egSCrIRFPFhYPDBUVj  
woAUhIj35UVgHAWH5+sIACfAAABP//7EELwMLAAGADwAFAC8AVUBSHRQCAQMPAQABDg0MCQQCAB  
wVAgQCBECaAGAEAAIEbQAGBwEDAQYDYAABAAACAQBgAAQFBQRUAQAQEBVgABQQFTBEQLismIxkXEB  
8RHxMTEggFFysBFA4BJjQ2HgEBFSE1NxcBJSEiBgCRFBY3ITI2JxE0JhcRFAYHISImNxE0NjchMh  
yBzt5aPj5aPgI8/06yWgEdAR78gwcKAQwGA30HDAEKUTQ1/ImkNgE0JQN9JTQCES0+AkJWQgQ6/v  
r6a7NZAR2hCgj9WgcMAQoIAqYiChL9WiU0ATYkAqYlNAE2AAEAAP9pBJsDUQAUB5AGwGAgABAU  
cIAQBEAABAHAAQAEMAUkcIwIFFisBFAYEjYInFwU+AT8BJjU0NiQgBBYEm57+8KB6cAL+myw2BA  
RqngEQAT4BEpwBgX7WfgEnA2s7hicmeJJ+1nx81gAAAAACAAD/nwOPAx0ABQA0AD5A0wQBAAIBRW  
MBAEQFAQIDAAMCAG0AAABuBAEBawMBUgQBAQEDWAADAQNMBWYAAAsKBg4HDgAFAAURBgUVKwkbIR  
EBERMyNi4CDgEWAYUCCv6N/fbMLEACPFw6BEIDHf32/owCCwFz/so+WD4CQlRCAEAAP+fAx8DHQ  
AMACNAIAkHAGEAAUcIAQFEAgEAAQBVAAEBZgEABgQADAEMAwUUKwEyFhAGJyInBzcmEDYBmaLk5K  
IqMrsBceYDHeT+vOYBDH3lcwFC5AAD//X/8gQgAssAGQAIACwANKAzAAEAwUBA2AABQAEAGUEYA  
YBAgAAALQGAQICAFgAAAIATBSaKyomJR8eGiIbIhwXBwUWKwEWBw4CBwYgJy4CJyY3PgI3NiAXHg  
IFMjY0JiIGFbY3FAYuAjY3MhYECYWBzZ8QXD+1XBafjQIFhYGNn5AcQEpcUB+Nv4HS2pql2pqtD  
xYPAJAKis8AXwdHgtGgixQUC2ASAodHgtGgCxSui1+SN9s12pql2y3Kz4C0l04BD4AAAQAAp9+A8  
ADPgAIACEAVQBjALNAFRMMAGQAjQECBCAcAgMCWlYCBQMER0uwDFBYQCYABAACAAQCbQACAwACA2  
sAAwUFA2MGAQAADegABQUBWQABAQ0BSRtLsBhQWEAnAAQAAGAEAm0AAgMAAGNrAAMFAAMFawYBAA  
AMSAAFBQFZAAEBDQFJG0AlBgEABABvAAQCBG8AAgMCbwADBQNvAAUBAQVUAAUFAVKAQUBTVLZQB  
MBAFlXSEc4NhkYBQQACAEIBWUUKwEyABAAIAAQAAE0JicGFx4BPwIWDgEXFjMeArcWBWYXNgEOAQ  
cyHwEeAhcWBhQWFRQWFRQWMzI2JjU0PgE3Ni4EiY4BBiY1ND4BNz4CNz4BAxYzMjcmBwYPAQYjDg  
EB4MgBGP7o/nL+5gEaAmCcFIBCBWQIBQWLC4WIj4cHgIKGBYkVv4ucK4oBhAcDBWUAgQkTBBI EA  
oCBhpeCBAOFDAiKAIQNBQihigICBIAgQqQkI+gGIaXBgpL0oCDBwDPv7m/nL+6AEYAY4BGv4ghN  
YcGAgmGgYMAhguQixAAkQgUDwsIHACHg6MaAIDAQYKAC+CojQUHFAEDFQsQAggVDgSIjYgGAoIBg  
IiHg4KIigKDg4SDAQa/PAURCwKAg8REAIYAAAAAIAAP++AsoDCwAFACIAMkAvFAUDAgQCAAFHaw  
ECAAJwBAEBAABABQAQAEAVgAAQKBWYFhIQBiIHIRAFBRUrASERAR8BEzIXHgEXERQGBWYjIi  
8BBWYjIicuATURNdy3NjMCg/3EAR4y7AcMdbMUARYSCg4bFPb2FBoNDBIWfHIMDQLD/UsBEi/jAv  
0FCB4U/TETIAcEEuzsEwUHIbMCzxMgBwUAAEAAP++AsoDCwAcACFAHg4BAQABRWMBAAEAbwIBAQ  
FmAQASEAwKABwBGwQFFCsBMhceARcRFAYHBiMiLwEHBiMiJy4BNRE0Njc2MwKKDAwTFAEWEgoOGx  
T29hQaDQwSFhYSDA0DCwUIHhT9MRMGbWQs7OwTBQcgEwLPEyAHBQAAAwAA//YD7QLGAawAGQAmAC  
xAKQAFAAQDBQRgAAMAAgEDAmAAQAAAVQAAQEAwAAAAQBMmZQzNDMyBgUaKzCUFjMhMjY0JiMhIg  
YTFBYzITI2NCYjISIGEXQWMyEyNjQmIyEiBkQqHgMZHioqHvznHSwBK4DGR4qKh785x0sASoeAx  
keKioe/OcdLD4eKio8KioBAh4qKjwqKgECHioqPCoqAAABAAAAQAEEVNluF8PPPUACwPoAAAAAN

hTrgIAAAAA2FOuAv/1/2kEmwNTAAACAAACAAAAAAAEAAANS/2oAAASb//X/9ASbAAEAAAAAAAA  
AAAAAAAAAAAAAAAAANA+gAAANN//0D6AAAAsoAAQv//8EmwAAA6AAAAMxAAAEff/1A8AAAAALKAAACyg  
AABDEAAAAAAAAAalgeAAUQBvgH2AjYCYgLGa7wEEARQBKIAAQAAAA0AZAAEAAAAAAAAACABAAIABzAA  
AAZgtwAAAAAAAAABIA3gABAAAAAAAAADUAAAABAAAAAAAAABAAgANQABAAAAAAAAACAACAPQABAAAAAA  
ADAAGARAABAAAAAAAAEAAGATAABAAAAAAFAAAsAVAABAAAAAAAGAAgAXwABAAAAAAAKACsAZwABAA  
AAAAALABMAkgaDAAEECQAAAGoApQADAAEECQABABABDwADAAEECQACAA4BHwADAAEECQADABABLQ  
ADAAEECQAEABABPQADAAEECQAFABYBTQADAAEECQAGABABYwADAAEECQAKAFYBcwADAAEECQALAC  
YByUNvcHlyawdodCAoQykMjAxOSBiesBvcmlnaW5hbCBhdXRob3JzIEAgZm9udGVsbG8uY29tZm  
9udGVsbG9SZWdlbGFyZm9udGVsbG9mb250ZWxsblZlcnNpb24gMS4wZm9udGVsbG9HZW5lcmF0ZW  
QgYnkgc3ZnMnR0ZiBmcm9tIEZvbnRlbGxvIHByb2plY3QuaHR0cDovL2ZvbnRlbGxvLmNvbQBDAG  
8AcAB5AHIAaQBnAGgAdAAgACgAQwApACAAMgAwADEAOQAgAGIAeQAgAG8AcgBpAGcAaQBuAGEAbA  
AgAGEAdQB0AGgAbwByAHMAIABAACAAZgBvAG4AdABlAGwAbABvAC4AYwBvAG0AZgBvAG4AdABlAG  
wAbABvAFIAZQBnAHUAbABhAHIAZgBvAG4AdABlAGwAbABvAGYAbwBuAHQAZQBzAGwAbwBwAGUAcg  
BzAGkAbwBuACAAMQAuADAAZgBvAG4AdABlAGwAbABvAECAZQBwAGUAcgBhAHQAZQBkACAAYgB5AC  
AAcWB2AGcAMgB0AHQAZgAgAGYAcgBvAG0AIABGAG8AbgB0AGUAbABsAG8AIABwAHIAbwBqAGUAYw  
B0AC4AaAB0AHQAcAA6AC8ALwBmAG8AbgB0AGUAbABsAG8ALgBjAG8AbQAAAAACAAAAAAAOAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA0BAGEDAQQBBQEGAQcBCAEJAQoBCwEMAQ0BDgAHdXAtaGFuZA  
hiZWxsLWFsdAR1c2VyB3BpY3R1cmULY29tbWVudC1hbHQDdGFuEgNvbW11bnQtaW52LWFsdDIDZX  
1lBWdsb2JlDmJvb2t0YXJrLWVtcHR5CGJvb2t0YXJrBG1lbnUAAAAAABAABH//wAPAAAAA  
AAAAAAAAAAAAAYABgAGAAYA1P/aQNT/2mwACwgsABVWEVZICBLuAAOUUwBlNaWLA0G7AoWWBmI  
pVWLACJWG5CAAIAGNjI2IbISGwAFmWAEMjRLIAAQBDYEItsAEssCBgZi2wAiwgZCCwFCwBCZasi  
gBCkNFY0VSWlghIyEbilggsFBQWCGwQFkbILa4UFghsDhZWSCxApQpDRWNFYWSwKFBYIbEBCkNFY0  
UgsDBQWCGwMFkbILDAUFggZiCKimEgsApQWGAAILAGUFghsApGyCwNlBYIbA2YBtgWVlZG7ABK1  
lZI7AAUFhLWVvtsAMsIEUgsAQlYWQgsAVDUFiWBSNCsAYjQhshIVmAWAtsAQsIyEjISBksQViQi  
CwBiNCsQEKK0VjsQEKK7ABYEVjsAMqISCwBkMgiICKsAersTAFJbAEJlFYFyFABYVJZWCNZISCwQF  
NYsAerGyGwQFkjsABQWGVZLbAFLLAHQyuyAAIAQ2BCLbAGLLAHl0IjILAAI0JhsAJiZrABY7ABYL  
AFKi2wBywgIEUgsAtDY7gEAGIGsABQWLBAYFlmsAFjYESwAWAtsAgssgcLAENFQiohsgABAENGQi  
2wCSyWAEMjRLIAAQBDYEItsAosICBFILABKyOwAEOWBCVgIEWKI2EgZCCwIFBYIbAAG7AwUFiWIB  
uwQFlZi7AAUFhLWbADJSNhrESwAWAtsAssICBFILABKyOwAEOWBCVgIEWKI2EgZLakUFiWABuwQF  
kjsABQWGVZsAMlI2FERLABYC2wDCwgsAAjQrILCgNFWCEBlyFZKiEtsA0ssQICRBkYUQtsA4ssA  
FgICCDENKsABQWCCwDCNCWbANQ0qWAFJYILANI0JZLbAPLCCwEGJmsAFjILgEAGOKI2GwDkNgII  
pgILAOI0IjLbAQLEtUWLEEZERZJLANZSN4LbARLEtRWetTWLEEZERZGyFZJLATZSN4LbASLLEAD0  
NVWLEPD0OwAWFCsA8rWbAAQ7ACJUKxDAILQrENAIvCsAEWIYcWwAyVQWLEBAENGsAQlQoqKIIoYyB  
AOKIEjsAFhIIoYyBaOKIEbsQEAQ2CwAiVCSAILyBaOKIFZsAxDR7ANQ0dgsAJiILAAUFiWQGBZzr  
ABYyCwC0NjuAQAYiCwAFBYsEBgWWawAWNGsQAAEYNesAFDsAA+sgEBAUNGQi2wEywAsQACRVRYsA  
8jQIBFsAsjQrAKI7ABYIEgYLABYbUQEAEADgBCQopgsRIGK7ByKxsiWS2wFCyXABMrLbAVLLEBEy  
stsBYssQITKy2wFyyXAxMrLbAYLLEEYstsBkssQUTKy2wGiYxBhMrLbAbLLEHEYstsBwssQgTKy  
2wHSyXCRMrLbAeLACwDSuxAAJFVfiwDyNCIEWwCyNCsAojSAFgQiBgsAFhtRAQAQAOAEJCimCxeg  
YrsHirGyJZLbAfLLEAHistsCAssQEeKy2wISyXh4rLbAiLLEDHistsCMssQeKy2wJCyXBR4rLb  
AlLLEGHistsCYssQceKy2wJyyxCB4rLbAoLLEJHistsCksIDyWAWAtsCosIGCwEGAGyQyOwAWBDsA  
ILyBAbYLApKiEtsCsssCorsCoqLbAsLCAGRyAgsAtDY7gEAGIGsABQWLBAYFlmsAFjYCNhOCMgil  
VYIEcgILALQ2O4BABiILAAUFiWQGBZzrABY2AjYTgbIVktsC0sALEAAkVUWLABFrAsKrABFTAbIL  
ktsC4sALANK7EAAkVUWLABFrAsKrABFTAbIlktsC8sIDWwAWAtsDAsALABRW04BABiILAAUFiWQG  
BZZrABY7ABK7ALQ2O4BABiILAAUFiWQGBZzrABY7ABK7AAFrQAAAAAEQ+IzixLwEVKi2wMSwgPC  
BHILALQ2O4BABiILAAUFiWQGBZzrABY2CwAENhOC2wMiwFzwtsDMsIDwGRyCwC0NjuAQAYiCwAF  
BYsEBgWWawAWNGsABDYbABQ2M4LbA0LLECABYLIC4gR7AAI0KwAiVJiophI0cjYSBYhshWbABI0  
KyMwEBFRQqLbAlLLAAFrAEJbAEJUcjRyNhsAlDK2WKLIMgIDyKOC2wNiywABawBCWwBCUgLkcjRy  
NhILAEI0KwCUMrILBgUFgggsEBRWLMCIAMg7MCJgMaWUJCiYcWCEMgiiNHI0cjYSNGYLAEQ7ACYI  
ChAFBYsEBgWWawAWNGsILABKyCKimEgsAJDYGQjsANDYWRQWLACQ2EbsANDYFmWwAmIgsABQWL  
BAYFlmsAFjYSMgILAEJiNGYTgbI7AIQ0awAiWwCENHI0cjYwAGsARDsAJiILAAUFiWQGBZzrABY2  
AjILABKyOwBENGsAersAULyBafJbACYiCwAFBYsEBgWWawAWOwBCZhilAEJWBki7ADJWBkUFghGy  
MhWSMgILAEJiNGYThZLbA3LLAAFiAgILAFJiAuRyNHI2EjPDgtsDgssAAWILAI0I0IGICBGi0ewAS  
sjYTgtsDkssAAWsAMlsAILRyNHI2GwAFRYLiA8IyEbsAilsAILRyNHI2EgsAulsAQlRyNHI2GwBi  
WwBSVJsAILybkIAAgAY2MjIFhiGyFZY7gEAGIGsABQWLBAYFlmsAFjYCMuIyAgPIo4IyFZLbA6LL  
AAFiCwCEMgLkcjRyNhIGCwIGBmsAJiILAAUFiWQGBZzrABYyMgIDyKOC2wOywJIC5GsAILrLJYID  
xZLrErARQrLbA8LCMgLKawAiVGUFggPFkusSsBFCstsD0sIyAuRrACJUzSWCA8WSMgLKawAiVGUF  
ggPFkusSsBFCstsD4ssDUrIyAuRrACJUzSWCA8WS6xKwEUKy2wPyywNiuKICA8sAQjQoo4IyAuRr

```

ACJUZSWCA8WS6xKwEUK7AEQy6wKystseAssAAWsAQlsAQmIC5HI0cjYbAJQysjIDwgLiM4sSsBFC
stseEessQgEJUKwABawBCWwBCUgLkcjRyNhILAEI0KwCUMrILBgUFggseBRWLMCIAMgG7MCJgMaWU
JCIyBHsARDsAJiILAAUfiwQGBZzrABY2AgsAerIIqKYSCwAkNgZCOWA0NhZFBYsAJDYRuWA0NgWb
ADJbACYiCwAFBYsEBGWWawAWNhsAilRmE4IyA8IzgbISAgRiNHsAerI2E4IVmxKwEUKy2wQiywNS
susSsBFCstseMssDYrISMgIDywBCNCIzixKwEUK7AEQy6wKystseQssAAVIEewACNCsgABARUUEy
6wMSotsEUssAAVIEewACNCsgABARUUEy6wMSotsEYssQABFBOWmiotsEcssDQqLbBILLAAfKujIC
4gRoojYTixKwEUKy2wSSywCCNCsEgrLbBKLLIAAEerLbBLLLIAAUerLbBMLLIBAEerLbBNLLIBAU
ErLbBOLLIAAEirLbBPLLIBAAUirLbBQLLIBAEirLbBRLLIBAUirLbBSLLIAAD4rLbBTLLIAAT4rLb
BULLIBAD4rLbBVLLIBAT4rLbBWLLIAAEarLbBXLLIAAUarLbBYLLIBAEarLbBZLLIBAUarLbBaLL
IAAEmrLbBbLLIAAUMrLbBcLLIBAEArLbBdLLIBAUmrLbBeLLIAAD8rLbBfLLIAAT8rLbBgLLIBAD
8rLbBhLLIBAT8rLbBiLLA3Ky6xKwEUKy2wYyywNyuwOystsGQssDcrsDwrLbBllLAAFrA3K7A9Ky
2wZiywOCsusSsBFCstseGcssDgrsDsrLbBoLLA4K7A8Ky2waSywOCuwPSstsGossDkrLrErARQrLb
BrLLA5K7A7Ky2wbCywOSuwPCstseG0ssDkrsD0rLbBuLLA6Ky6xKwEUKy2wbyywOiuwOystsHAssD
orsDwrLbBxLLA6K7A9Ky2wciyzCQQA0VYIRsjIVlCK7AIZbADJFB4sAEVMC0AS7gAyFJYsQEBjl
mwAbkIAAgAY3CxAAVCsgABACqxAAVCswoCAQgqsQAFQrMOAAEIKrEABkK6AsAAAQAJKrEAB0K6AE
AAAQAJKrEADAESxJAGIUviwQIhYsQNkRLEmAYhRWLoIgAABBECIYlRYsQMARFlZWVmzDAIBDCq4Af
+FsasNsQIARAAA') format('truetype');
}
#sidebar-menu .icon-up-hand:before {
 content: '\e801';
}
#sidebar-menu .icon-bell-alt:before {
 content: '\e802';
}
#sidebar-menu .icon-user:before {
 content: '\e803';
}
#sidebar-menu .icon-picture:before {
 content: '\e804';
}
#sidebar-menu .icon-comment-alt:before {
 content: '\e805';
}
#sidebar-menu .icon-tag:before {
 content: '\e806';
}
#sidebar-menu .icon-comment-inv-alt2:before {
 content: '\e807';
}
#sidebar-menu .icon-eye:before {
 content: '\e808';
}
#sidebar-menu .icon-globe:before {
 content: '\e809';
}
#sidebar-menu .icon-bookmark-empty:before {
 content: '\e810';
}
#sidebar-menu .icon-bookmark:before {
 content: '\e811';
}
#header .icon-menu:before {
 content: '\e812';
}
#sidebar-menu .icon,
#header #hamburger.icon-menu {
 font-family: 'fontello';
}

```



```

 }
 #sidebar-menu .e-menu-icon::before {
 color: #656a70;
 }
 /*icon styles */
</style>
</body>
</html>

```

## INDEX.CSHTML

```

<div class="content">
 <div> Responsive Sidebar using Content Template</div>
</div>
<style>
 .content {
 margin-top: 15vw;
 text-align: center;
 font-size: 32px;
 color: #1784c7;
 }
</style>

```

Output be like the below.



## Hide Sidebar

The following example demonstrates how to hide layout page sidebar. Initially sidebar is rendered with layout page. While navigate to another page, it hides the layout page sidebar.

## CSHTML

```

<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>@ViewBag.Title - My ASP.NET Application</title>
 @Styles.Render("~/Content/css")
 @Scripts.Render("~/bundles/modernizr")

```

```

<!-- Syncfusion Essential JS 2 Styles -->
<link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css" />
<!-- Syncfusion Essential JS 2 Scripts -->
<script src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
</head>
<body>
 <div class="navbar navbar-inverse navbar-fixed-top">
 <div class="container">
 <div class="navbar-header">
 <button type="button" class="navbar-toggle" data-
toggle="collapse" data-target=".navbar-collapse">

 </button>
 @Html.ActionLink("Application name", "Index", "Home", new {
area = "" }, new { @class = "navbar-brand" })
 </div>
 <div class="navbar-collapse collapse">
 <ul class="nav navbar-nav">
 @Html.ActionLink("Home", "Index", "Home")
 @Html.ActionLink("About", "About", "Home")
 @Html.ActionLink("Contact", "Contact", "Home")

 </div>
 </div>
 </div>
 @{Html.EJS().Sidebar("sidebar-menu").Width("220px").Target(".main-
content").ContentTemplate(@<div class="dock-menu">
 <!-- normal state element declaration -->
 <div class="sidebar-header header-cover">
 <div style="padding-top:30px">
 <div class="sub-title">
 Layout Page Sidebar
 </div>
 </div>
 </div>
 <!-- end of normal state element declaration -->
 </div>).Render();}
 <div id="maintext" class=".main-content">
 <div>
 @RenderBody()
 </div>
 </div>
 @Html.EJS().ScriptManager()
 @Scripts.Render("~/bundles/jquery")
 @Scripts.Render("~/bundles/bootstrap")
 @RenderSection("scripts", required: false)
 <style>
 #sidebar-menu {
 padding-top: 30px;
 background-color: #1694CA
 }
 .sub-title {
 text-align: center;
 font-size: 20px;

```

```

 padding: 3vw;
 color: white;
 }
 .container {
 margin-left: 10px;
 }
</style>
</body>
</html>

```

## CONTACT.CSHTML

```

@{Html.EJS().Sidebar("sidebar").Width("230px").Target(".maincontent").ContentTemplate(@<div>
 <!-- Sidebar content -->
 <div class="sidebar-header header-cover">
 <div style="padding-top:30px">
 <div class="sub-title">
 Contact Page Sidebar
 </div>
 </div>
 </div>
</div>).Render();}
<div>
 <h2 class="sidebar-heading"> Responsive Sidebar With Treeview</h2>
 <p class="paragraph-content">
 This is a graphical aid for visualising and categorising the
 site, in the style of an expandable and
 collapsable treeview component. It auto-expands to display the
 node(s), if any, corresponding to the currently
 viewed title, highlighting that node(s) and its ancestors. Load-
 on-demand when expanding nodes is available
 where supported (most graphical browsers), falling back to a
 full-page reload. MediaWiki-supported caching,
 aside from squid, has been considered so that unnecessary re-
 downloads of content are avoided where possible.
 The complete expanded/collapsed state of the treeview persists
 across page views in most situations.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading">Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
 enim ad minim veniam, quis
 nostrud exercitation ullamco laboris nisi ut aliquip ex ea
 commodo consequat. Duis aute irure
 dolor in reprehenderit in voluptate velit esse cillum dolore eu
 fugiat nulla pariatur.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod

```

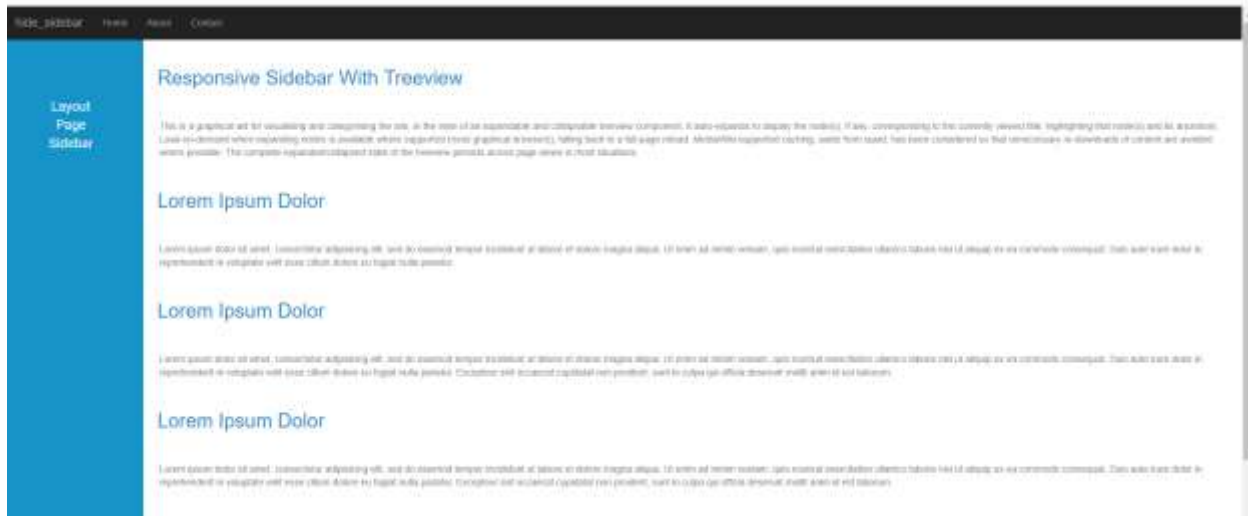
```

 tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
 minim veniam, quis nostrud
 exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat. Duis aute irure dolor in
 reprehenderit in voluptate velit esse cillum dolore eu fugiat
 nulla pariatur. Excepteur sint
 occaecat cupidatat non proident, sunt in culpa qui officia
 deserunt mollit anim id est laborum.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod
 tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
 minim veniam, quis nostrud
 exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat. Duis aute irure dolor in
 reprehenderit in voluptate velit esse cillum dolore eu fugiat
 nulla pariatur. Excepteur sint
 occaecat cupidatat non proident, sunt in culpa qui officia
 deserunt mollit anim id est laborum.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod
 tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
 minim veniam, quis nostrud
 exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat. Duis aute irure dolor in
 reprehenderit in voluptate velit esse cillum dolore eu fugiat
 nulla pariatur. Excepteur sint
 occaecat cupidatat non proident, sunt in culpa qui officia
 deserunt mollit anim id est laborum.
 </p>
 <div class="line"></div>
 <h2 class="sidebar-heading"> Lorem Ipsum Dolor</h2>
 <p class="paragraph-content">
 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
 eiusmod
 tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
 minim veniam, quis nostrud
 exercitation ullamco laboris nisi ut aliquip ex ea commodo
 consequat. Duis aute irure dolor in
 reprehenderit in voluptate velit esse cillum dolore eu fugiat
 nulla pariatur. Excepteur sint
 occaecat cupidatat non proident, sunt in culpa qui officia
 deserunt mollit anim id est laborum.
 </p>
</div>
<script>
 document.addEventListener('DOMContentLoaded', function () {
 var sidebarTreeview = document.getElementById("sidebar-
 menu").ej2_instances[0];
 // Toggle the Sidebar

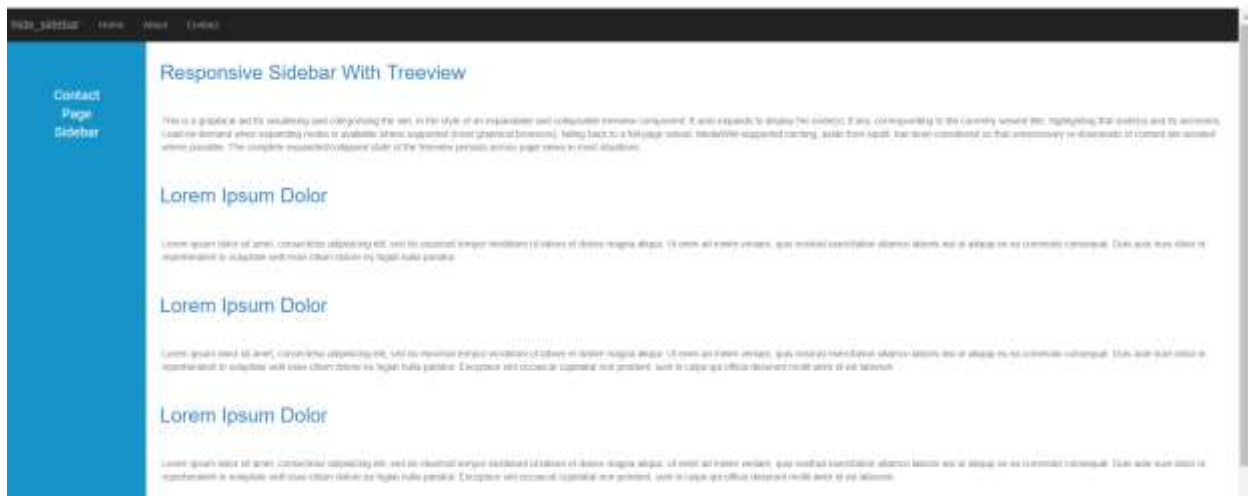
```

```
 if (sidebarTreeview.isOpen) {
 sidebarTreeview.hide();
 }
 });
</script>
<style>
 #sidebar {
 padding-top: 30px;
 background-color: #1694CA;
 height:100%;
 position:fixed;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 color: white;
 }
 .sidebar-heading {
 color: #1c86c8;
 margin: 40px 0;
 padding: 2px;
 }
 .paragraph-content {
 font-family: 'Poppins', sans-serif;
 padding: 5px;
 font-weight: 300;
 color: grey;
 }
 body {
 padding-top: 10px;
 }
 .e-content-animation {
 padding-top: 1.5vw;
 }
 #myCarousel {
 padding-left: 1vw;
 }
</style>
```

Output be like the below in layout page.



Output be like the below, while navigate to other page, it hides the layout page sidebar.



## Sidebar with partial view

The following example demonstrates how to render the sidebar with partial view. Sidebar element is placed inside the `RenderPartialView.cshtml` and refer that sidebar element in layout page.

### CSHTML

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-
scale=1.0">
 <title>@ViewBag.Title - My ASP.NET Application</title>
 <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
 @Styles.Render("~/Content/css")
 @Styles.Render("~/Content/styles.css")
 @Scripts.Render("~/bundles/modernizr")
```

```

 <link rel="stylesheet"
href="https://cdn.syncfusion.com/ej2/material.css" />
 <script
src="https://cdn.syncfusion.com/ej2/dist/ej2.min.js"></script>
 </head>
 <body>
 @Html.Partial("_RenderPartialView")
 <div id="body_content" class="maincontent">
 <div style="padding: 1.5vw 3vw;">
 @RenderBody()
 </div>
 </div>
 @Html.EJS().ScriptManager()
 @Scripts.Render("~/bundles/jquery")
 @Scripts.Render("~/bundles/bootstrap")
 @RenderSection("scripts", required: false)
 </body>
</html>

```

### TEXTBOX.CSHTML

```

@using Syncfusion.EJ2
<table class="table">
 <div id="top-bar">
 <div class="component">
 Components
 <i class="fa fa-angle-right" style="font-size:16px"></i>
 Inputs
 <i class="fa fa-angle-right" style="font-size:16px"></i>
 TextBox
 </div>
 </div>
 <div id="feature">
 <div class="content">
 <div class="body-inner">
 TextBox
 </div>
 <div style="text-align:left">
 <div style="font-size:2vw">Overview</div>
 <div class="text">The ASP.NET MVC TextBox (text field)
control is most useful for editing, displaying, or entering plain text on
forms to capture user names, phone numbers, email, and more. This control is
an extended version of the HTML5 TextBox (input type text) control with
icons, floating labels, different sizing, grouping, validation states, and
more. It is available in an HTML5/CSS version and an ASP.NET MVC
version.</div>
 </div>
 <div style="padding:2vw 0 2vw 0;text-align:left">
 <div style="font-size:1.5vw">Key Features</div>
 <div class="text">
 <p>1. Floating Label : Floats the placeholder
text while focus.</p>
 <p>2. Input Group : Group the icons, buttons
along with textbox.</p>

```

```

 <p>3. Validation States: Provides styles for
success, error, and warning states.</p>
 <p>4. Multiline : Handles multiline input with
placeholder text.</p>
 </div>
</div>
<div class="link">
 <div>
 <a
href="https://ej2.syncfusion.com/aspnetmvc/documentation/textbox/getting-
started">Documentation Link
 <a
href="https://ej2.syncfusion.com/aspnetmvc/TextBoxes/DefaultFunctionalities#
/material">Samples Link
 </div>
</div>
</div>
</div>
</table>

```

### RENDERPARTIALVIEW\_MVC.CSHTML

```

<!-- Applpicable for MVC only -->
@using Syncfusion.EJ2.Navigations;
@{
 ViewBag.Title = "_RenderPartialView";
}
@{Html.EJS().Sidebar("sidebar-
menu").Width("290px").Target(".maincontent").MediaQuery("(min-
width:670px)").ContentTemplate(@<div>
 <!-- Sidebar content -->
 <div class="sidebar-header header-cover" style="background-color:
#1694CA">
 <div style="text-align:center">
 <div class="sidebar-brand">
 Essential JS 2 Syncfusion Components
 </div>
 </div>
 <div style="padding:0 5px 10px 5px">
 <div class="searchbox">
 <input id="search-by" type="text" placeholder="Search
Components">
 <i class="fa fa-search"
style="font-size:20px"></i>
 </div>
 </div>
 </div>
 <div class="control-section">
 <div class="control_wrapper accordion-control-section">

@Html.EJS().Accordion("accordion").ExpandMode(ExpandMode.Single).EnablePersi
stence(true).Items(new List<AccordionAccordionItem>
 {
 new AccordionAccordionItem { Header =
"Navigations",Expanded=true, Content = "#navigation" },

```



```

 new AccordionsAccordionItem { Header = "Layouts", Content =
"#layouts" },
 new AccordionsAccordionItem { Header = "Inputs", Content =
"#inputs" },
 }).Render()
</div>
<ul id="navigation" style="display:none">
 Sidebar
 ContextMenu
 Accordion

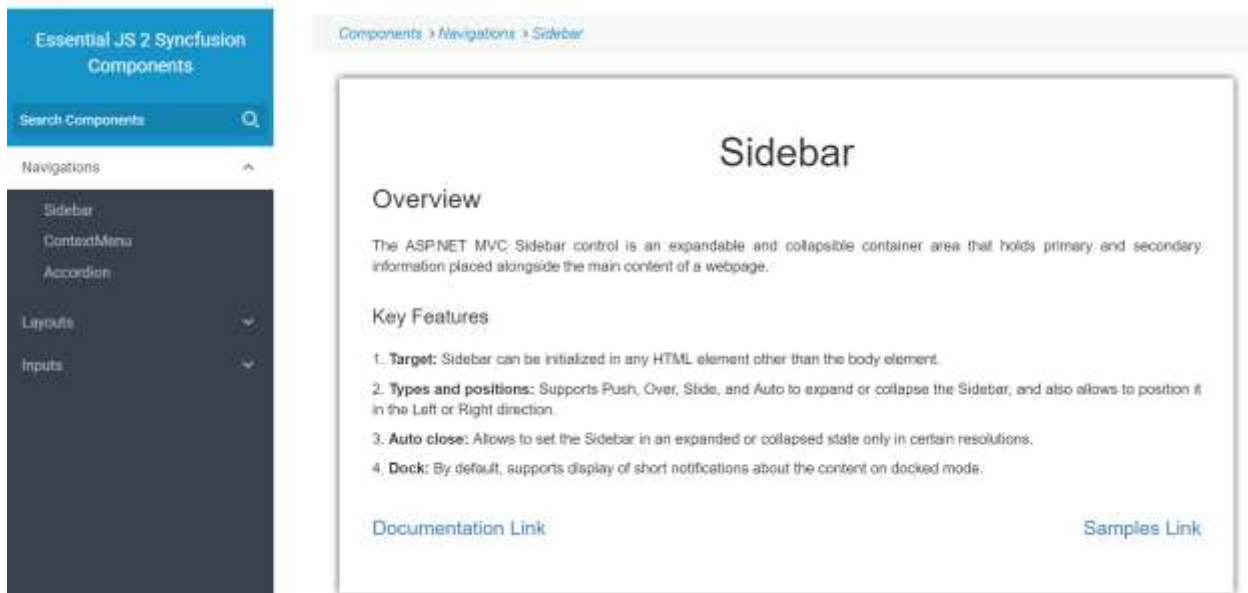
<ul id="layouts" style="display:none">
 Avatar
 Card
 Splitter

<ul id="inputs" style="display:none">
 Input Mask
 TextBox

</div>
<!-- end of normal state element declaration -->
</div>).Render();

```

Output be like the below in layout page.



### Adding Dropdownlist inside the sidebar

While clicking the dropdownlist items inside the sidebar element, it closes the sidebar component. By default, the `closeonDocumentClick` property will close the Sidebar element whenever the click action will be triggered outside of the Sidebar element. The `DropDownList` popup element placed outside of the Sidebar element (body tag instead of sidebar element). So, during the popup element interactions (click, mousedown) the `closeonDocumentClick` will trigger and close the sidebar.

To overcome this behavior in your application, you can add one common class by using `cssClass` property to all the components that has appended their elements outside of the sidebar component. Based on that class you can prevent the Sidebar close action by using close event.

The following example shows how to prevent the closes of sidebar while click the dropdownlist.

### CSHTML

```
<!-- sidebar element declaration -->
@{Html.EJS().Sidebar("default-
sidebar").Width("280px").CloseOnDocumentClick(true).ContentTemplate(@<div>
 <div class="title-header">
 <div style="display:inline-block"> Dropdownlist </div>
 </div>

@Html.EJS().DropDownList("games").DataSource((IEnumerable<object>)ViewBag.da
ta).CssClass("custom_class").Render()
</div>).Close("onClose").Render();
}
<!-- end of sidebar element -->
<!-- Main Content declaration -->
<div>
 <div class="title default">Main content</div>
 <div class="sub-title">
 Place your primary content here.
 </div>
</div>
<script>
 // Prevent the sidebar element from closing.
 function onClose() {
 if(document.activeElement.classList.contains("custom_class")){
 args.cancel = true;
 }
 }
</script>
<style>
 /* sidebar element styles */
 #default-sidebar {
 background-color: rgb(25, 118, 210);
 color: #ffffff;
 }
 .title {
 text-align: center;
 font-size: 20px;
 padding: 15px;
 }
 .sub-title {
 text-align: center;
 font-size: 16px;
 padding: 10px;
 }
</style>
```

### SIDEBAR-DROPDOWNLIST.CS

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication.Controllers
{
 public partial class SidebarController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```

Output be like the below.



## Signature

### Getting Started with ASP.NET MVC Signature Control

This section briefly explains about how to include [ASP.NET MVC Signature](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

## PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
```

```
</body>
```

### Add ASP.NET MVC Signature control

Now, add the Syncfusion ASP.NET MVC Signature control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
<div class='wrap'>
 @Html.EJS().Signature("signature").Render()
</div>
<style>
 .wrap {
 margin: 0 auto;
 width: 300px;
 text-align: center;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
</style>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Signature control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

### Customization of Signature Control

The Signature control draws stroke/path using `moveTo()` and `lineTo()` methods to connect one or more points while drawing in canvas. The stroke width can be modified by using its color and width. And the background can be modified by using its background color and background image.

#### Stroke Width

The variable stroke width is based on the values of [MaxStrokeWidth](#), [MinStrokeWidth](#) and [Velocity](#) for smoother and realistic signature. The default value of minimum stroke width is set as 0.5, maximum stroke width is set as 2.5 and velocity is set as 0.7.

In the following example, minimum stroke width is set as 0.5, maximum stroke width is set as 3 and velocity is set as 0.7.

### CSHTML

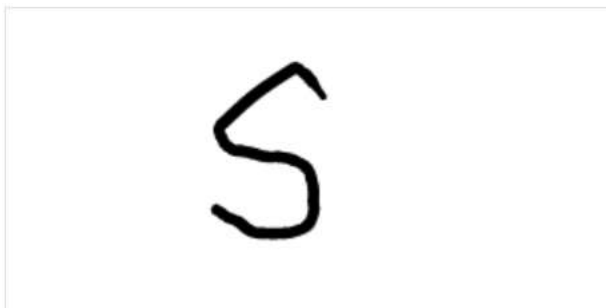
```
<div class='wrap'>

@Html.EJS().Signature("signature").MaxStrokeWidth(3).MinStrokeWidth(0.5).Velocity(0.7).Render()
</div>
<style>
 .wrap {
 margin: 0 auto;
 width: 300px;
 text-align: center;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
</style>
```

### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

Output be like the below.



### Stroke Color

Color of the stroke can be specified by using [StrokeColor](#) property and it accepts hexadecimal code, RGB, and text. The default value of this property is "#000000".

### CSHTML

```
<div class='wrap'>
 <div id='input'>
 <input type='text' id='text' placeholder='Enter the Stroke Color Value'>
 @Html.EJS().Button("btn").Content("Set Stroke Color").IsPrimary(true).Render()
 </div>
```

```
<div id="signature-control">
 @Html.EJS().Signature("signature").Render()
</div>
</div>
<script>
 document.getElementById("btn").addEventListener('click', function () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var color = document.getElementById("text").value;
 signature.strokeStyle = color;
 });
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
 #input {
 margin-bottom: 30px;
 }
 #text {
 height: 30px;
 width: 300px;
 }
</style>
```

#### **DEFAULT.CS**

```
public ActionResult Default()
{
 return View();
}
```

Output be like the below.



### Background Color

Background color of a signature can be specified by using [BackgroundColor](#) property and it accepts hexadecimal code, RGB, and text. The default value of this property is “#ffffff”.

### CSHTML

```
<div class='wrap'>
 <div id="input">
 <input type="text" id="text" placeholder="Enter the Background Color Value">
 @Html.EJS().Button("btn").Content("Set Background Color").IsPrimary(true).Render()
 </div>
 <div id="signature-control">
 @Html.EJS().Signature("signature").Render()
 </div>
</div>
<script>
 document.getElementById("btn").addEventListener('click', function () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var bgColor = document.getElementById("text").value;
 signature.backgroundColor = bgColor;
 });
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
 #input {
 margin-bottom: 30px;
 }
 #text {
```



```

 height: 30px;
 width: 300px;
 }
</style>

```


### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

Output be like the below.



### Background Image

Background image of a signature can be specified by using [BackgroundImage](#) property. The background image can be set by either hosting the image in our local IIS or online image.

### CSHTML

```

<div class='wrap'>
 <div id="input">
 <input type="text" id="text" placeholder="Enter the URL of the
background Image">
 @Html.EJS().Button("btn").Content("Set Background
Image").IsPrimary(true).Render()
 </div>
 <div id="signature-control">
 @Html.EJS().Signature("signature").Render()
 </div>
</div>
<script>

```

```
document.getElementById("btn").addEventListener('click', function () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var bgImage = document.getElementById("text").value;
 signature.backgroundImage = bgImage;
});
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 text-align: center;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
 #input {
 margin-bottom: 30px;
 }
 #text {
 height: 30px;
 width: 350px;
 }
</style>
```

#### **DEFAULT.CS**

```
public ActionResult Default()
{
 return View();
}
```

Output be like the below.

Set Background Image

See Also

- [Save with Background](#)

### Open and Save Signature

The Signature control supports to open the signature by using hosted/online URL or base64. And it also supports various save options like image, base64, and blob.

### Open Signature

The signature control opens a pre-drawn signature as either base64 or hosted/ online URL using the `load` method. It supports the PNG, JPEG, and SVG image's base64.

### CSHTML

```
<div class='wrap'>
 <div id="input">
```

```

<input type="text" id="text" placeholder="Enter the Base64 or URL of
signature">
 @Html.EJS().Button("btn").Content("Open").IsPrimary(true).Render()
</div>
<div id="signature-control">
 @Html.EJS().Signature("signature").Render()
</div>
</div>
<script>
 document.getElementById("btn").addEventListener('click', function () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var sign = document.getElementById("text").value;
 signature.load(sign);
 });
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 text-align: center;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
 #input {
 margin-bottom: 30px;
 }
 #text {
 height: 30px;
 width: 350px;
 }
</style>

```

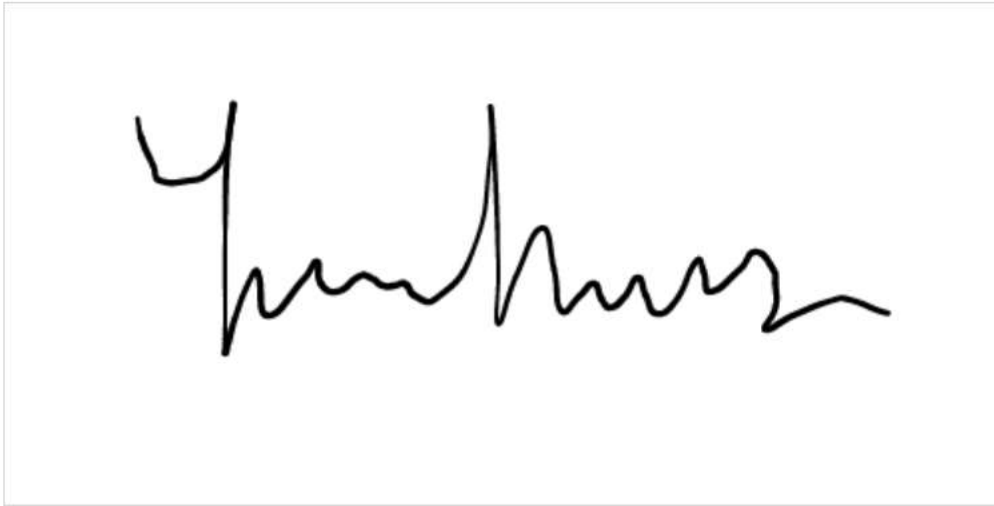
#### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

Output be like the below.

Open

### Save Signature

The Signature control saves the signature as base64, blob, and image like PNG, JPEG, and SVG.

#### Save as Base64

The `getSignature` method is used to get the signature as base64 with the PNG, JPEG, and SVG type. This can be loaded to signature using `load` method.

### CSHTML

```
<div class='wrap'>
 <h4>Sign here</h4>
 <div id="signature-control">
 @Html.EJS().Signature("signature").Render()
 </div>
 @Html.EJS().Button("btn").Content("Save as
Base64").IsPrimary(true).Render()
 @Html.EJS().Dialog("dialog").Width("80%").Header("Base64 of the
signature").Visible(false).ShowCloseIcon(true).Render()
</div>
<script>
 document.getElementById("btn").addEventListener('click', function () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var dialogObj =
ej.base.getInstance(document.getElementById("dialog"), ej.popups.Dialog);
 dialogObj.content = signature.getSignature();
 dialogObj.show();
 });
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 }
</style>
```

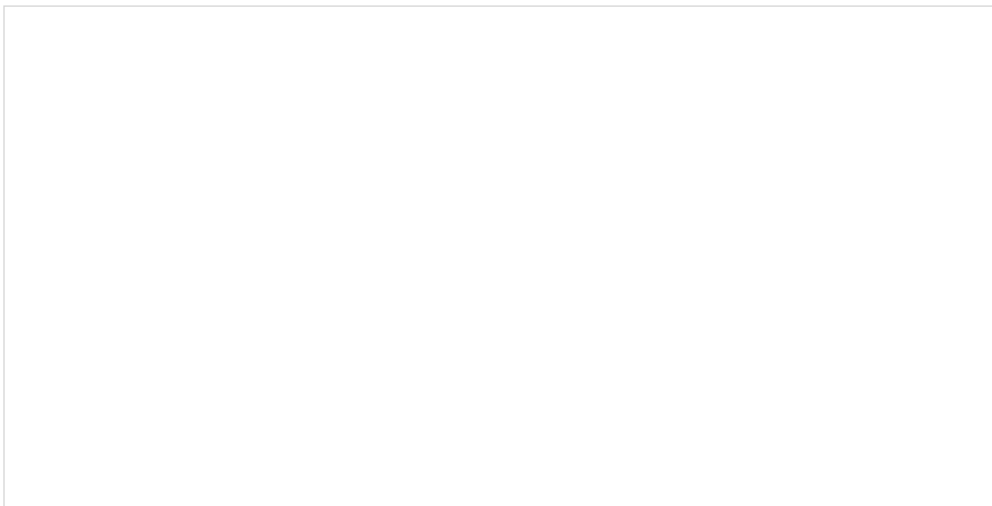
```
#signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
}
#btn {
 margin-top: 30px;
}
</style>
```

### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

Output be like the below.

## Sign here



Save as Base64

### *Save as Blob*

The `saveAsBlob` method is used to save the signature as a Blob. It is defined as the chunk of binary data stored as a single entity in a database system.

### *Save as Image*

The `save` method is used to save the signature as an image. And it accepts file name and file type as parameter. The file type parameter supports PNG, JPEG, and SVG and the default file type is PNG.

### CSHTML

```
<div class='wrap'>
```

```
<div>
 Sign here
 @Html.EJS().SplitButton("btn").IconCss("e-sign-icons e-
save").Content("Save").Items(ViewBag.datasource).Select("onSelect").Render()
</div>
<div id="signature-control">
 @Html.EJS().Signature("signature").Render()
</div>
</div>
<script>
function onSelect(args) {
 var signature =
document.getElementById("signature").ej2_instances[0];
 signature.save(args.item.text, "Signature");
}
</script>
<style>
.wrap {
 margin: 0 auto;
 width: 500px;
}
#signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
}
.e-split-btn-wrapper {
 float: right;
 margin-bottom: 5px;
 margin-right: 50px;
}
@@font-face {
 font-family: 'font-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj1tSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6Wka0AAADQAAAAANmhoZWEIUQQLAAAArAAAACRobXR4KAAAAAAYAAAAA
obG9jYQowB4oAAAH4AAAAFm1heHABIAEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCId8
AAA3IAAAAJwABAAAEAAAAAFwEAAAAAAD9AABAAAAAIAAAACgABAAAAQAAC7rwy18
PPPUACwQAAAAAN3B814AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAIAAAAKAXgADAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCGQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAQAAEAAAAABAAAAAQAQAAEAAAAABAAAAAIAAADAAAAFAADAAEAAAAUAAQAOGA
AAAYABAABAAALnAecK//8AAOcA5wT//wAAAAAQAQAGAAgAAAAABAAIAAwAEAAUABgAHAAGACQAAAA
AAAA6AFoAiACyAOgCKAPQBFYEtgAAAAQAAAAA/QD8wADAAsAGQAJAAABESERARUzNTMVITUjESE
RMxUzESMRIREjESMRFSERIZUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
X1/7iAR5I/O4BZv6aA1r8pkcDWUhHAAAAwAAAAAD7gP0AAMABwAPAAALFSE1EzM1IwEhESMRIRE
jA0T9d1p8fP78A96r/XKl8WVlAgP//Bkd6P7OATIAAAMAAAAA/QD9AACAAAYAGQAANYUnNxcBJzc
HFz8DNS8HDWIMASTqO+kB0+qpbulyBQQCAGQFpggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc
EAWEBAWQAAAAABAAAAAD9APqAAIABgAKAA8ACUHNyUBJwElByc3AQm1CQEBN8ctAj/+laMBbAF
PeaF6/XNQAVsCjf78nyzH+v6ZowFkC3ihd/3r/qxJAoABCwAAAGAAAAAD8wPoAB4AIgAAEW8HFR8
KMz8DFSE1IQE3CQI9BgSJBwCEAwICAwQHBwLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg00Dg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkaAgAAAAAD8wPqAEkBGgAAAR8FDwwVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEVBtUvFg8TA4M
GBAMCAQEBAQQHBAKQCcDAwECAQEDAwQFBgYHCACJCAkICQkJCAkIBwgHBgYFBAMDAGECBQUHCQk
KDAwNDQ4ODw4dHB0iJv4aJgQCBAYGCAKLDA8ICAKJCgoLDaKkCQkJCAkIEA4ODQwLCQkHBgUEEW
MCAgcGBQUDAwIBAgIDBAQEUBGgYHBwCHBgGBgYFBQUEAwMCAgEBAQICBAUFBgCHBQIDAgMDJgc
HBwCHBgYGCwsJBwv+oAMCAQEBAUHChEVGRwVfHYWfXyGhXwTEBAODQUGBAUDAVwHBgUCAQIDBAU
```

```
DpAMEBgKcW0PCAkJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVfwwMDQ40CwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBgJCwwOGfEVFRMSEhAQDg4NDAsKCgk
IDwsKCglwChgODg8ODw4NDAoFBAMDawEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQBQYGBgg
ICaKHBwcGBgYFBQQEBAACAgEBAgIDBAQEBAQUGBgYHBwcICAgHBwcGBRoafBUXDA0NKAcFBAQCAgE
BAgQEB/6gBAQFBgYNDxASEh4gISEXfHYUEXIYFBAIBwQCAgICBAFbCQsNBggHCacICAgEoxgdHh4
eHh0cGgwMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAUAAAAA/QD5AA5AI4
AswDaAXcAAAEzHw8VDwcvBj0BLxUlHxMDDwUvFz8BHwk/BTUVDDclHwclLws1PwyfBicXDwQvCzU
/DTmfAycPDh8KDwQdAR8XOWE/CBMfAx0Bhw07AT8NPQIvLCMPAQMxBwgODg0LCwoJCACGBQQDagE
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwcICQgKGxwcHh8V/sMGBg8SExUXFfxkgIB8fHx4dHBs
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAWEBbiAaGxwdHR4eCagIBwYGAgIBAgMFBsa
fHh0bGhgfwAEXBi0dIh8afg/+lyAZHQ0LCQgHBQDQADagEDAwQJDhERExUXGBoc6QUJCQCgGfxMPDg0
LCgkHBgUDAgEDBAMFBQYHCAGJDxAREhAQIYs5Dw4NDAsJCACFAwMDAQEBAgMGCAoLDRcaExh0BAM
CAQICAwQJDA8RExUXFxoAGhIkJCMhIR8ODg4MCwsQDwkHBgYElwMDAgEBAgMEBAUFBgCHBwgICAK
ICQgHCACGBgYFBAMDAGICAgMFBQYGCAGKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwcGxoAGRg
XFxYWFBQCTAEDBAcICgsMDQ0PDxAQEBfTBAUDAwMCAQEBAQIDAwMFBDCsIASWfgoKCgCHBgUFBAM
CBQQGCQsIqQoKExQTFBITERUSEA8NCwkGBQIF/ncEawQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRi
QCgcGvxowFBUTFBISAwEBAwUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVfHUbDQ0LDAoKCgg
HBgUFAwIEAgEBAwQGBwoLWwEFBwgH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSEhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTEYWNwgJCAgHBwYGBgUEAwMCAgICAwMEBQYGBgCHAgJCG0RERAQEBApDw4ODQw
LCgkKEwwNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAgHBQUEAwEBAgMADAAAAAAD8gP0AAgADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRWGFp8JHRCZIIiLOIhkDBAYICgoGBgc
FDAoKCAyFAgEDBAYICQsGBgYGDAAoKCAyFAgFxDxYGrBMPHEgWBgoBEPyuRAPk/BwBr96cVT+yGUs
DIhMWBxwcChYQLgcGBgYGCYJBgsLCQgHBQEBAQEBCAChCgsFBwYGCwsJCACFAQEBCQYICQsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAA/QDqAAGADYAPQBBAABNxmVITUBJRUFCTsBPwk9AS8
KDwoLEQMHAwERAYERIQQJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBwYFBgoIBwQCAQEg7YL1/vY9A+j8GAfBqf7tQpYBR3oHBgYMCgkHAWICAQECAgMHCQoMBgYHBwY
GDAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAAACgABAAEAAAAAAAAIABwALAAEAAAAAAAAAMACgASAAEAAAAAAAAAQACgAcAAEAAAAAAU
ACwAmAAEAAAAAAAAAYACgAxAAEAAAAAAAAoALAA7AAEAAAAAAAAAEgBnAAMAAQQJAAAAAgB5AAMAAQ
JAAEAFAB7AAMAAQQJAAIADgCPAAMAAQQJAAAFACdAAMAAQQJAAQAFACxAMAAQQJAAUAFgDFAAM
AAQQJAAAYAFADbAAMAAQQJAAoAWADvAAMAAQQJAAAsAJAFHIGZvbnQtaWNvbnNSZWdlbGFyZm9udC1
pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVYXRlZCBlc2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBpAGM
AbwBuAHMAUGBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBJAG8AbgBzAGYAbwBuAHQALQBpAGMAbwB
uAHMAVGBlAHIAcWBPAG8AbgAgADEALgAwAGYAbwBuAHQALQBpAGMAbwBuAHMARgBvAG4AdAAgAGc
AZQBwAGUAcgBhAHQAZQBkACAAdQBsAGkAbgBnACAuUwB5AG4AYwBmAHUAcWBPAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdABlAGQAaQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcWBPAG8AbgAuAGMAbwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBAEFAQYBBwEIAQkBCgELAAd
zYXZlXzAyB3NhdmUtMDEHfWZWRpdF8wMwdlZG10XzAxBNwSfWfYDHBhaW50LWJlY2tldA9wYWludC1
idWNrZXQtd2YGAw1hZ2VzC3BpY3RlcmVzLXdmAAAA) format('trueType');
font-weight: normal;
font-style: normal;
}
.e-sign-icons {
font-family: 'font-icons' !important;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-save::before {
```




```
 content: '\e701';
 }
</style>
```

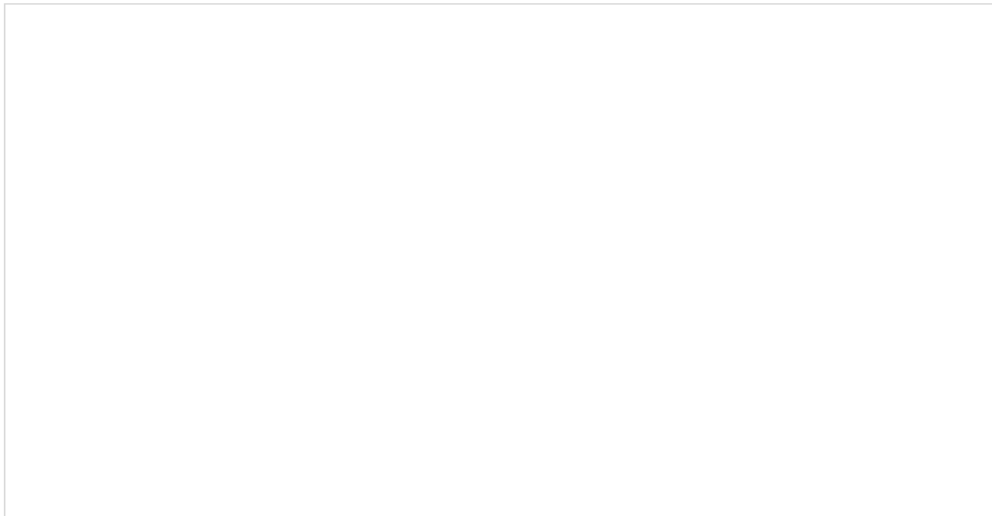
### DEFAULT.CS

```
public ActionResult Default()
{
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Png"
 });
 items.Add(new
 {
 text = "Jpeg"
 });
 items.Add(new
 {
 text = "Svg"
 });
 ViewBag.datasource = items;
 return View();
}
```

Output be like the below.

Sign here

 Save ▼



### Save with Background

The [saveWithBackground](#) property is used to save the signature with its background and its default value is true. So, by default the signature is saved with its background.

In the following sample, the background color is set as 'rgb(103 58 183)' and save with background as true.

**CSHTML**

```

<div class='wrap'>
 <div>
 Sign here
 @Html.EJS().SplitButton("btn").IconCss("e-sign-icons e-
save").Content("Save").Items(ViewBag.datasource).Select("onSelect").Render()
 </div>
 <div id="signature-control">
 @Html.EJS().Signature("signature").BackgroundColor("rgb(103 58
183)").Render()
 </div>
</div>
<script>
 function onSelect(args) {
 var signature =
document.getElementById("signature").ej2_instances[0];
 signature.save(args.item.text, "Signature");
 }
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
 .e-split-btn-wrapper {
 float: right;
 margin-bottom: 5px;
 margin-right: 50px;
 }
 @@font-face {
 font-family: 'font-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6Wka0AAADQAAAAANmhoZWEIUQQLAAAArAAAACRobXR4KAAAAAAYAAAAA
obG9jYQowB4oAAAH4AAAAFm1heHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCI8
AAA3IAAAAJwABAAAEAAAAAFwEAAAAAAD9AABAAAAAIAAAACgABAAAAQAAC7rwy18
PPPUACwQAAAAAAN3B814AAAAA3cHyXgAAAAAD9AP0AAACACAAAAAIAAAAKAXgADAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCGQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAA
EAAAABAAAAQAQAAEAAAABAAAAQAQAAEAAAABAAAAAIAAADAAAAFAADAAEAAAAUAAQA0gA
AAAYABAABAAALnAecK//8AA0cA5wT//wAAAAAAQAGAAgAAAABAAIAAwAEAAUABgAHAAgACQAAAAA
AAAA6AFoAiACyAOgCKAPQBFYETgAAAAQAAAAA/QD8wADAAsAGQAJAAABESERARUzNTMVITUjESE
RMxUzESMRIREjESMRFSERIZUjNSEDHV3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFx/uIBHgI8j4/
X1/7iAR5I/O4BZv6aA1r8pkcDWUHHAAAAwAAAAAD7gP0AAMABwAPAAAlFSE1EzM1IwEhESMRIRE
jA0T9d1p8fP78A96r/XKl8WVlAgP/BkD6P7OATIAAAMAAAAA/QD9AACAAyAGQAANYUnNxcBJzc
HFz8DNS8HDwIMASTqO+kB0+qpbulyBQQCAGQFpggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc
EAWEBAwQAAAAABAAAAAAD9APqAAIABgAKAA8AACUHNyUBJwElByc3AQm1CQEBN8ctAj/+laMBbAF
PeaF6/XNQAVScjf78nyzH+v6ZowFkC3ihd/3r/qxJAoABCwAAAQAAAAAD8wPoAB4AIgAAEW8HFR8
KMz8DFSE1IQE3CQI9BgSJBwcEAWICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AfH+xwGwBg00Dg8
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkAAgAAAAAD8wPqAEkBGgAAAR8FDwwVHxM/CjU
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEVBtUvFg8TA4M

```

```

GBAMCAQEBAQQHHBAKCQCDAwECAQEDAwQFBgYHCAcJCAkICQkJCAkIBwgHBgYFBAMDAgECBQUHCQk
KDAwNDQ4ODw4dHB0iJv4aJgQCBAYGCAkLDA8ICAKJCgoLDAkKCQkJCAkIEA4ODQwLCQkHBgUEEwM
CAgcGBQUDAwIBAgIDBAQEUBgYHBwCgHBgcGBgYFBQUEAwMCAgEBAQICBAUFBgCHBQIDAQMDJgc
HBwCGBwYGCwsJBwv+oAMCAQEBAUHChEVGRwVfHxYWFxYgHxwTEBAODQUGBAUDAVwHBgUCAQIDBAU
DpAMEBggKCw0PCAKJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0
tGhMVfwwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBggJCwwOGfEVFRMSEhAQDg4NDAsKCGk
IDwsKCGlwChgODg8ODw4NDAoFBAMDAwEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQPBBQYGBgg
ICAKHBwCGBgYFBQQEBAWCAgEBAgIDBAQEUBgYHBwCICAgHBwCGBRoAFBUXDA0NKACFBAQCAGe
BAgQEB/6gBAQFBgYNDxASEh4gISEXFhYUEXiYFBAIBwQCAgICBAFbCQsNBggHCACICAgEoxgdHh4
eHh0cGgwMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAUAAAAA/QD5AA5AI4
AswDaAXcAAAEzHw8VDwcvBj0BLxULHxMDDwUvFz8BHwk/BTUVDDclHwclLws1PwYfBicXDwQvCzU
/DTMFAYcPDh8KDwQdAR8XOWE/CBMfAx0Bhw07AT8NPQIVLCMPAQMxBwgODg0LCwoJCAcGBQQDAgE
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwCICQgKGxwcHh8V/sMGBg8SExUXFkgIB8fHx4dHBs
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAWEBbiAaGxwdHR4eCAGIBwYGAgIBAgMFBsA
fHh0bGhgfwAEXBi0dIh8aFg/+1yAZHQ0LCQgHBQQDAgEDAwQJDhERExUXGBoc6QUJCQCgGfxMPDg0
LCgkHBgUDAgEDBAMFBQYHCAgJDxAREhAQIyS5Dw4NDAsJCAcFAwMDAQEBAgMGCAoLDRcaExh0BAM
CAQICAwQJDA8RExUXFxoagHikJCMhIR8ODg4MCwsQDwkHBgYE1wMDAgEBAgMEBAUFBgCHBwGICAK
ICQgHCACGBgYFBAMDAgICAgMFBQYGCAGKCgoMDBIJBwgKCwsNDQ4QDyMkJEMdHhwdHBwCgxoAGRg
XFxYWFbQCTAEDBACICgsMDQ0PDxAQEBfTBAUDAwMCAQEBAQIDAwMFBDCsIASWFgoKCgCHBgUFBAM
CBQQGCQsIQQoKExQTFBITERUSEA8NCwkGBQIF/nCEAwQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFRi
QCgcGvxowFBUTFBISAwEBawUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVFhUbDQ0LDAoKCgg
HBgUFAwIEAgEBawQGBwoLWwEFBwgH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU
HBwkJCgsICAgJCAkSExMTFBuUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSHhEQChMQDQwIBgIBAQI
EBgUFBgUGAXQTEYWNwgJCAgHBwYGBgUEAwMCAgICAwMEBQYGBgCHCAgJCG0RERAQEBApDw4ODQw
LCgkKEwWNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAgHBQUEAwEBAGMADAAAAAAD8gP0AAGADAAQABQAGAA
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRrWGFp8JHRCZiIiLOIhkBAYICgoGBgc
FDAoKCAyFAgEDBAYICQsGBgYGDAAoKCAyFAgFxDxYGRBMPHEgWBgoBEPYURAPk/BwBr96cVT+yGUs
DIhMWBxwcChYQLgcGBgYGCChYJBgsLCQgHBQEBAQEBCACHCgsFBwYGCwsJCAcFAQEBCQYICQsGBj8
QHAYGHw8WI1H9BQL7/GMD6AAAAAQAAAAA/QDqAAGADYAPQBBAABNxmVITUBJRUFCTsBPwk9AS8
KDwoLEQMHAwERAYERIJJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY
GBwYFBgoIBwQCAQEg7YL1/vY9A+j8GAfBqf7tQpYBR3oHBgYMCgkHAWICAQECAgMHCQoMBgYHBwY
GDAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAEgDeAAEAAAAAAAAAAQA
AAAEAAAAAAAAEACgABAAEAAAAAAAAIABwALAAEAAAAAAAAAMACgASAAEAAAAAAAAAQACgAcAAEAAAAAAAU
ACwAmAAEAAAAAAAYACgAxAAEAAAAAAAOALAA7AAEAAAAAAASAEgBnAAMAAQQJAAAAAgB5AAMAAQQ
JAAEFAB7AAMAAQQJAAIADgCPAAMAAQQJAAAMAFACdAAMAAQQJAAQAFACxAMAAQQJAAUAFgDFAAM
AAQQJAAAYAFADbAAMAAQQJAAoAWADvAAMAAQQJAAASAJAFHIGZvbnQtaWNvbnNSZWdlbGFyZm9udC1
pY29uc2ZvbnQtaWNvbnNWZXJzaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVYXlRlZCB1c2luZyB
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBPAGM
AbwBuAHMAUGBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBJAG8AbgBzAGYAbwBuAHQALQBPAGMAbwB
uAHMAVGBlAHIAcWBPAG8AbgAgADEALgAwAGYAbwBuAHQALQBPAGMAbwBuAHMARgBvAG4AdAAgAGc
AZQBwAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcWBPAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdABlAGQAaQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcWBPAG8AbgAuAGMAbwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBAEFAQYBBwEIAQkBCgELAAd
zYXZlXzAyB3NhdmUtMDEHfWZWRpdF8wMwdlZG10XzAxBNwSfWfYDHBhaW50LWJlY2tldA9wYWludC1
idWNrZXQtd2YGAw1hZ2VzC3BpY3RlcmVzLXdMAAAA) format('trueType');
 font-weight: normal;
 font-style: normal;
}
.e-sign-icons {
 font-family: 'font-icons' !important;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;

```

```
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-save::before {
 content: '\e701';
}
</style>
```

### DEFAULT.CS

```
public ActionResult Default()
{
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Png"
 });
 items.Add(new
 {
 text = "Jpeg"
 });
 items.Add(new
 {
 text = "Svg"
 });
 ViewBag.datasource = items;
 return View();
}
```

Output be like the below.

Sign here

 Save ▼



## Draw a Signature

### Draw

The `draw` method is used to draw a text as signature with different font families like Arial, Serif, with different font sizes. It accepts text, font family, font size as its parameters. The default font family is "Arial", and the default font size is "30".

### CSHTML

```
<div class='wrap'>
 <div id="input">
 <table>
 <tbody>
 <tr>
 <td><div>Enter the Text:</div></td>
 <td><input type="text" id="text" placeholder="Enter the
Text"></td>
 </tr>
 <tr>
 <td style="padding-top:10px"><div>Font
Family:</div></td>
 <td style="padding-top:10px">
 @Html.EJS().DropDownList("font").Placeholder("Select
a font").Value("Arial").PopupHeight("200px").DataSource(
 (IEnumerable<object>)ViewBag.data).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "Text"
}).Render()
 </td>
 </tr>
 <tr>
 <td style="padding-top:10px"><div>Font Size:</div></td>
 <td style="padding-top:10px">
 @Html.EJS().DropDownList("size").Placeholder("Select
a size").Value("20").PopupHeight("200px").DataSource(
 (IEnumerable<object>)ViewBag.size).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Value = "Text"
}).Render()
 </td>
 </tr>
 </tbody>
 </table>

 @Html.EJS().Button("btn").Content("Draw").IsPrimary(true).Render()
 </div>
 <div id="signature-control">
 @Html.EJS().Signature("signature").Render()
 </div>
</div>
<script>
 document.getElementById("btn").addEventListener('click', function () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var text = document.getElementById("text").value;
 var font = document.getElementById("font").ej2_instances[0].value;
 var size = document.getElementById("size").ej2_instances[0].value;
 signature.draw(text, font, size);
 });
</script>
```

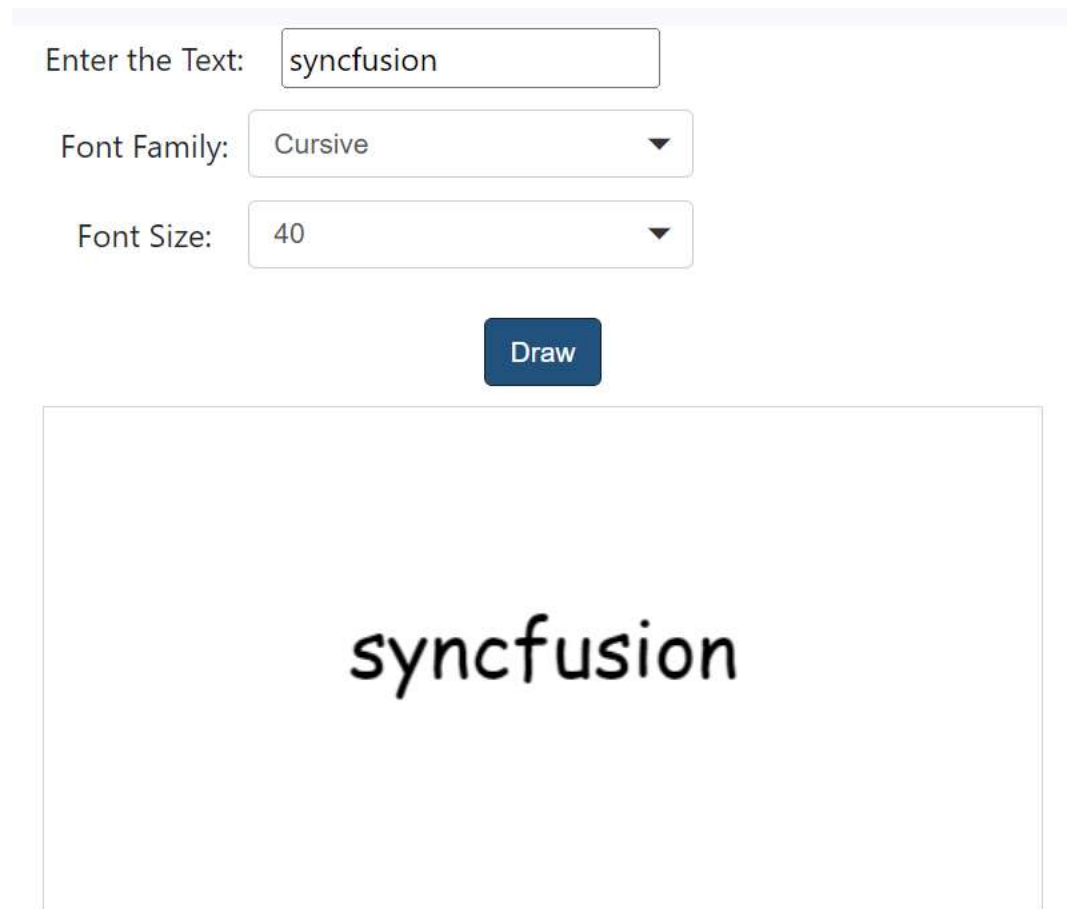
```
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;
 text-align: center;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
</style>
```

### DEFAULT.CS

```
public ActionResult Default()
{
 List<object> fontData = new List<object>();
 fontData.Add(new { Text = "Arial" });
 fontData.Add(new { Text = "Serif" });
 fontData.Add(new { Text = "Sans-serif" });
 fontData.Add(new { Text = "Cursive" });
 fontData.Add(new { Text = "Fantasy" });
 ViewBag.data = fontData;
 List<object> sizeData = new List<object>();
 sizeData.Add(new { Text = "20" });
 sizeData.Add(new { Text = "30" });
 sizeData.Add(new { Text = "40" });
 sizeData.Add(new { Text = "50" });
 ViewBag.size = sizeData;

 return View();
}
```

Output be like the below.



The screenshot displays the Signature application's user interface. At the top, there is a light purple header bar. Below it, the 'Enter the Text:' label is followed by a text input field containing 'syncfusion'. Below this, the 'Font Family:' label is followed by a dropdown menu showing 'Cursive'. The 'Font Size:' label is followed by a dropdown menu showing '40'. A blue 'Draw' button is positioned below the font settings. At the bottom, a large white canvas displays the word 'syncfusion' in a cursive font.

### User Interactions

The below interactions were available in Signature, and we can walk through one by one.

- Undo and Redo
- Clear
- Disabled
- ReadOnly

### Undo and Redo

In the Signature, every action can be maintained as a snap for undo and redo operations. And maintained `SnapIndex` for indexing the snap collection.

The `undo` method reverts the last action of signature by decreasing `SnapIndex` value to index previous snap. Here, `canUndo` method is used to ensure whether undo can be performed or not.

The `redo` method reverts the last undo action of the signature by increasing the `SnapIndex` to get the next snap. Here, `canRedo` method is used to ensure whether redo can be performed or not.

### Clear

The `clear` method is used to clears the signature and makes the canvas empty. This is also considered in Undo/ Redo. Here, `isEmpty` method is used to ensure whether the signature is empty or not.

### Disabled

The `disabled` property is used to enables/disables the signature control. In the disabled state, the user is not allowed to draw signature. And it can't be focused until the user enabled the signature.

### ReadOnly

The `isReadOnly` property is used to enables/disables the ReadOnly Signature. It can be focused but it prevents drawing in Signature.

The following sample explains about user interactions available in signature.

### CSHTML

```
<div class='wrap'>
 <div id="option">
 <table>
 <tr>
 <td>

@Html.EJS().Button("undoBtn").Content("Undo").IsPrimary(true).Disabled(true)
.Render()

 </td>
 <td>

@Html.EJS().Button("redoBtn").Content("Redo").IsPrimary(true).Disabled(true)
.Render()

 </td>
 <td>

@Html.EJS().Button("clearBtn").Content("Clear").IsPrimary(true).Disabled(true)
.Render()

 </td>
 <td>
 <div style="margin-bottom: 5px; margin-left: 200px;">

@Html.EJS().CheckBox("disable").Label("Disabled").Change("disableChange").Render()

 </div>
 <div style="margin-left: 200px;">

@Html.EJS().CheckBox("readonly").Label("ReadOnly").Change("readOnlyChange").Render()

 </div>
 </td>
 </tr>
 </table>
 </div>
 <div id="signature-control">
 @Html.EJS().Signature("signature").Change("change").Render()
 </div>
</div>
<script>
 document.getElementById("undoBtn").addEventListener('click', function ()
 {
 var signature =
document.getElementById("signature").ej2_instances[0];
 if (!signature.isReadOnly && !signature.disabled) {
 signature.undo();
 }
 });
</script>
```



```

 }
 });
 document.getElementById("redoBtn").addEventListener('click', function ()
 {
 var signature =
document.getElementById("signature").ej2_instances[0];
 if (!signature.isReadOnly && !signature.disabled) {
 signature.redo();
 }
 });
 document.getElementById("clearBtn").addEventListener('click', function
 () {
 var signature =
document.getElementById("signature").ej2_instances[0];
 if (!signature.isReadOnly && !signature.disabled) {
 signature.clear();
 }
 });
 function disableChange(args) {
 var signature =
document.getElementById("signature").ej2_instances[0];
 signature.disabled = args.checked;
 }
 function readOnlyChange(args) {
 var signature =
document.getElementById("signature").ej2_instances[0];
 signature.isReadOnly = args.checked;
 }
 function change() {
 var signature =
document.getElementById("signature").ej2_instances[0];
 var undoBtn = document.getElementById("undoBtn").ej2_instances[0];
 var redoBtn = document.getElementById("redoBtn").ej2_instances[0];
 var clearBtn = document.getElementById("clearBtn").ej2_instances[0];
 if (!signature.disabled && !signature.isReadOnly) {
 if (signature.canUndo()) {
 undoBtn.disabled = false;
 } else {
 undoBtn.disabled = true;
 }
 if (signature.canRedo()) {
 redoBtn.disabled = false;
 } else {
 redoBtn.disabled = true;
 }
 if (!signature.isEmpty()) {
 clearBtn.disabled = false;
 } else {
 clearBtn.disabled = true;
 }
 }
 }
}
</script>
<style>
 .wrap {
 margin: 0 auto;
 width: 500px;

```

```
 text-align: center;
 }
 #signature {
 border: 1px solid lightgray;
 height: 100%;
 width: 100%;
 }
 #option {
 margin-bottom: 10px;
 }
</style>
```

#### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

Output be like the below.

Undo Redo Clear

☐ Disable

☐ ReadOnly

#### Accessibility in Signature Component

The Signature component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Signature component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support | Not Applicable |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
```

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### Keyboard interaction

The Signature component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Signature component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + Z** | **Undo the last action.** |

| **Ctrl + Y** | **Redo the last action.** |

| **Ctrl + S** | **To save the signature.** |

| **delete** | **Erases all the signature strokes signed by user.** |

### Ensuring accessibility

The Signature component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Signature component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Signature component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

### How To

#### Integration with Toolbar

The Signature control integrates with the toolbar and the interaction performed using the `change` event of the toolbar. In that, `canUndo`, `canRedo` and `isEmpty` methods were used to enable/disable undo, redo, and clear buttons.

#### CSHTML

```
<div class="control-section">
 <div id="signature-toolbar-control">
 @(Html.EJS().Toolbar("toolbar").Width("100%").Items(new
List<Syncfusion.EJ2.Navigations.ToolbarItem> {
 new Syncfusion.EJ2.Navigations.ToolbarItem { Text = "Undo",
PrefixIcon = "e-icons e-undo", TooltipText = "Undo (Ctrl + Z)" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Text = "Redo",
PrefixIcon = "e-icons e-redo", TooltipText = "Redo (Ctrl + Y)" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Type =
Syncfusion.EJ2.Navigations.ItemType.Separator },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Type =
Syncfusion.EJ2.Navigations.ItemType.Button, Template = "<button id='save-
option'></button>", TooltipText = "Save (Ctrl + S)" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Type =
Syncfusion.EJ2.Navigations.ItemType.Separator },
 new Syncfusion.EJ2.Navigations.ToolbarItem { TooltipText =
"Stroke Color", Type = Syncfusion.EJ2.Navigations.ItemType.Input, Template =
"<input id='stroke-color' type='color'>" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Type =
Syncfusion.EJ2.Navigations.ItemType.Separator },
 new Syncfusion.EJ2.Navigations.ToolbarItem { TooltipText =
"Background Color", Type = Syncfusion.EJ2.Navigations.ItemType.Input,
Template = "<input id='bg-color' type='color'>" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Type =
Syncfusion.EJ2.Navigations.ItemType.Separator },
 new Syncfusion.EJ2.Navigations.ToolbarItem { TooltipText =
"Stroke Width", Type = Syncfusion.EJ2.Navigations.ItemType.Input, Template =
"<input id='stroke-width' type='text'>" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Type =
Syncfusion.EJ2.Navigations.ItemType.Separator },
 new Syncfusion.EJ2.Navigations.ToolbarItem { Text = "Clear",
PrefixIcon = "e-sign-icons e-clear", TooltipText = "Clear" },
 new Syncfusion.EJ2.Navigations.ToolbarItem { TooltipText =
"Disabled", Align = Syncfusion.EJ2.Navigations.ItemAlign.Right, Type =
```

```

Syncfusion.EJ2.Navigations.ItemType.Input, Template = "<input id='disabled'
type='checkbox' />"
 })
 .Created("Created")
 .Clicked("Clicked")
 .Render()
)
 <div id="signature-control">

@Html.EJS().Signature("signature").MaxStrokeWidth(2).Change("signChange").Re
nder();
 </div>
</div>
</div>
}
<script>
 function signChange(args) {
 if (!signature.isEmpty()) {
 var saveBtn =
ej.base.getComponent(document.getElementById("save-option"), 'split-btn');
 clearButton();
 saveBtn.disabled = false;
 }
 updateUndoRedo();
 }
 function Created(args) {
 new ej.buttons.CheckBox({
 checked: false,
 label: "Disabled",
 change: (args) => {
 signature.disabled = args.checked;
 }
 }, "#disabled");
 var items = [{ text: 'Png' }, { text: 'Jpeg' }, { text: 'Svg' }];
 new ej.splitbuttons.SplitButton({ content: 'Save', iconCss: 'e-sign-
icons e-save', items: items, select: onSelect, disabled: true }, '#save-
option');
 var strokeColor = new ej.inputs.ColorPicker({
 modeSwitcher: false,
 columns: 4,
 presetColors: {
 'custom': ['#000000', '#e91e63', '#9c27b0', '#673ab7',
'#2196f3', '#03a9f4', '#00bcd4',
'#009688', '#8bc34a', '#cddc39', '#ffeb3b', '#ffc107']
 },
 beforeTileRender: (args) => {
 args.element.classList.add('e-circle-palette');
 args.element.appendChild(ej.base.createElement('span', {
className: 'e-circle-selection' }));
 },
 showButtons: false, mode: 'Palette', cssClass: 'e-stroke-color',
change: strokeColorChanged
 });
 strokeColor.appendTo('#stroke-color');
 var bgColor = new ej.inputs.ColorPicker({
 noColor: true,
 modeSwitcher: false,

```

```

 columns: 4,
 presetColors: {
 'custom': ['#ffffff', '#f44336', '#e91e63', '#9c27b0',
 '#673ab7', '#2196f3', '#03a9f4', '#00bcd4',
 '#009688', '#8bc34a', '#cddc39', '#ffeb3b']
 },
 beforeTileRender: (args) => {
 args.element.classList.add('e-circle-palette');
 args.element.appendChild(ej.base.createElement('span', {
 className: 'e-circle-selection' }));
 },
 showButtons: false, mode: 'Palette', cssClass: 'e-bg-color',
 change: bgColorChanged
 });
 bgColor.appendTo('#bg-color');
 var ddl = new ej.dropdowns.DropDownList({
 dataSource: [1, 2, 3, 4, 5],
 width: '60',
 value: 2,
 change: function (args) {
 signature.maxStrokeWidth = args.value;
 }
 });
 ddl.appendTo('#stroke-width');

 ej.base.addClass([strokeColor.element.nextElementSibling.querySelector('.e-
selected-color')], 'e-sign-icons');

 ej.base.addClass([bgColor.element.nextElementSibling.querySelector('.e-
selected-color')], 'e-sign-icons');
 clearButton(true);
 var toolbarItems = document.querySelectorAll('.e-toolbar .e-toolbar-
items .e-toolbar-item .e-tbar-btn.e-tbtn-txt');
 for (var i = 0; i < toolbarItems.length; i++) {
 if (toolbarItems[i].children[0].classList.contains('e-undo')) {
 var undoButton = ej.base.getComponent(toolbarItems[i],
 'btn');
 undoButton.disabled = true;
 }
 if (toolbarItems[i].children[0].classList.contains('e-redo')) {
 var redoButton = ej.base.getComponent(toolbarItems[i],
 'btn');
 redoButton.disabled = true;
 }
 }
 function Clicked(args) {
 var saveBtn = ej.base.getComponent(document.getElementById("save-
option"), 'split-btn');
 if (signature.disabled && args.item.tooltipText != 'Disabled') {
 return;
 }
 switch (args.item.tooltipText) {
 case 'Undo (Ctrl + Z)':
 if (signature.canUndo()) {
 signature.undo();
 updateUndoRedo();
 }
 }
 }
 }

```

```

 updateSaveBtn();
 }
 break;
case 'Redo (Ctrl + Y)':
 if (signature.canRedo()) {
 signature.redo();
 updateUndoRedo();
 updateSaveBtn();
 }
 break;
case 'Clear':
 signature.clear();
 if (signature.isEmpty()) {
 clearButton();
 saveBtn.disabled = true;
 }
 break;
}
}
function updateSaveBtn() {
 var saveBtn = ej.base.getComponent(document.getElementById("save-
option"), 'split-btn');
 if (signature.isEmpty()) {
 saveBtn.disabled = true;
 }
}
function updateUndoRedo() {
 var undoButton; var redoButton;
 var tlItems = document.querySelectorAll('.e-toolbar .e-toolbar-items
.e-toolbar-item .e-tbar-btn.e-tbtn-txt');
 for (var i = 0; i < tlItems.length; i++) {
 if (tlItems[i].children[0].classList.contains('e-undo')) {
 undoButton = ej.base.getComponent(tlItems[i], 'btn');
 }
 if (tlItems[i].children[0].classList.contains('e-redo')) {
 redoButton = ej.base.getComponent(tlItems[i], 'btn');
 }
 }
 if (signature.canUndo()) {
 undoButton.disabled = false;
 } else {
 undoButton.disabled = true;
 }
 if (signature.canRedo()) {
 redoButton.disabled = false;
 } else {
 redoButton.disabled = true;
 }
}
function onSelect(args) {
 signature.save(args.item.text, 'Signature');
}
function strokeColorChanged(args) {
 if (signature.disabled) {
 return;
 }
}

```

```

 var selElem = this.element.nextElementSibling.querySelector('.e-
selected-color');
 selElem.style.borderBottomColor = args.currentValue.rgba;
 signature.strokeColor = args.currentValue.rgba;
 }
 function bgColorChanged(args) {
 if (signature.disabled) {
 return;
 }
 var selElem = this.element.nextElementSibling.querySelector('.e-
selected-color');
 signature.backgroundColor = args.currentValue.rgba;
 selElem.style.borderBottomColor = args.currentValue.rgba;
 }
 function clearButton(isCreated) {
 var tlItems = document.querySelectorAll('.e-toolbar .e-toolbar-items
.e-toolbar-item .e-tbar-btn.e-tbtn-txt');
 for (var i = 0; i < tlItems.length; i++) {
 if (tlItems[i].children[0].classList.contains('e-clear')) {
 var clrBtn = ej.base.getComponent(tlItems[i], 'btn');
 if (isCreated || signature.isEmpty()) {
 clrBtn.disabled = true;
 } else {
 clrBtn.disabled = false;
 }
 }
 }
 }
}
</script>
@section Scripts {
 <script>
 var signature =
ej.base.getComponent(document.getElementById('signature'), 'signature');
 document.getElementById('save-option').addEventListener('click',
function () {
 signature.save();
 });
 var items = [
 {
 text: 'Png'
 },
 {
 text: 'Jpeg'
 },
 {
 text: 'Svg'
 }
];
 </script>
}
<style>
 @font-face {
 font-family: 'font-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSfwAAAEoAAAAVmNtYXDOQM6IAAABqAAAAE5nbHlmPRF
AxQAAAhAAAAlsaGVhZB6Wka0AAADQAAAAANmhoZWElUQQLAAAArAAAACRobXR4KAAAAAAYAAAAA
obG9jYQowB4oAAAH4AAAAFmlheHABIAGEAAABCAAAACBuYW1lbLYTYgAAC3wAAAJJcG9zdIlCId8

```



AAA3IAAAjwABAAAEAAAAFwEAAAAAAD9AABAAAAAAGCgABAAAAQAAC7rwy18  
PPPUACwQAAAAAN3B814AAAAA3cHyXgAAAAAD9AP0AAAACAACAAAAAEEAAAKAXgADAAAAA  
AAgAAAAoACgAAP8AAAAAQAQAAZABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnCGQAAAAAXAQAAAAAABAAAAAABAAAAQA  
AAAAAABAAAAQAQAAEAAAABAAAAQAQAAEAAAABAAAAAIAAADAAAAFAADAAEAAAAUAAQOgA  
AAAYABAABALnAecK//8AA0cA5wT//wAAAAAQAAGAAgAAAAABAAIAAwAEAAUABgAHAAGACQAAAA  
AAAA6AFoAiACyAOgCKAPQBFYEtGAAAAQAAAAA/QD8wADAAsAGQAJAABESERARUzNTMVITUjESE  
RMxUzESMRIREjESMRFSERIzUjNSEDHv3EAR5HSP6bSAH0j0dH/TZIRwPoR0j8pwFfx/IBHgI8j4/  
X1/7iAR5I/O4BZv6aA1r8pkcDWUHHAAAAwAAAAAD7gP0AAMABwAPAAAFSE1EzM1IwEhESMRIRE  
jA0T9d1p8fP78A96r/XK18WV1AgP//Bkd6P7OATIAAAMAAAAA/QD9AACAAyAGQAANYUnNxcBJzc  
HFz8DNS8HDwIMASTqO+kB0+qpbulyBQQCAGQFPggJCQoJCQkMOuo66QHS6alu6XIICQoJCgkIpgc  
EAwEBAwQAAAAABAAAAAD9APqAAIABgAKAA8ACUHNyUBJwElByc3AQMLCQEBN8ctAj/+lAMBbAF  
PeaF6/XNQAVsCjf78nyZH+v6ZowFkC3ihd/3r/qxJAoABCwAAgAAAAAD8wPoAB4AIgAAEW8HFR8  
KMz8DFSE1IQE3CQI9BgSJBwCEAwICAwQHBwkLqgkJCQoJCQlGAo39iP7IPwE6AFH+xwGwBg0ODg8  
PDxAQDxAPDw4ODaoGBAICBAZGM0gBOT7+xwHyATkAAgAAAAAD8wPqAEkBGgAAAR8FDwVHxM/CjU  
vFCUzNT8RHxMVJx8BFQcfBh0BDw0rAS8OPwo1LwsjDwQBDwMVHxU7AT8DAT8EPQEvBTUvFg8TA4M  
GBAMCAQEBAQQHBAKQCcDAwECAQEDAwQFBgYHCACJCAkICQkJCAkIBwgHBgYFBAMDAGECBQUHCQk  
KDAWNDQ4ODw4dHB0iJv4aJgQCBAYGCAKLDA8ICAKJCgoLDAKQCkJCAkIEA4ODQwLCQkHBgUEEW  
CAGcGBQUDAwIBAgIDBAQEUBGgYHBwCHBgCGBgYFBQUEAwMCAgEBAQICBAUFBgCHBQIDAQMDJgc  
HBwCGBwYGCwsJBwv+oAMCAQEBAUHChEVGRvVfHYWfXyGhXwTEBAODQUGBAUDAVVHBgUCAQIDBAU  
DpAMEBggKCw0PCAKJCQkKCwsLCwwMDQ0NDQ0MCwsLCgkJCRAODAsJCAYFAwIB3AYFBgYGBgYGDQ0  
tGhMVFWwMDQ4OCwsLCwoLCgoJCQgIBwYGBAQDAQEBAgMEBggJCwwOGfEVFRMSEhAQDg4NDAsKCgk  
IDwsKCglwChgODg8ODw4NDaoFBAMDawEBAQEBAgMDBQUFDRARExQWFxgYGBkYBhMdGBQBPQYGBgg  
ICAKHBwCGBgYFBQQEBAWCAgEBAgIDBAQEUBGgYHBwCICAGHBwCGBRoAFBUXDA0NKACFBAQCAG  
BAGQEB/6gBAQFBgYNDxASEh4gISEXFhYUEXiyFBAIBwQCAgICBAFbCQsNBggHCAcICAgEoxgdHh4  
eHh0cGgWMCwsKCQgIBwYFAwMCAQEBAgIDBAQFBgYMDxARERISEhIRAAAAAUAAAAA/QD5AA5AI4  
AswDaAXCAAAEzHw8VDwcvBj0BLxUlHxMDDwUvFz8BHwk/BTUVDDclHwclLws1PwyfBicXDwQvCzU  
/DTmfAycPDh8KDwQdAR8XOWE/CBMfAx0BHw07AT8NPQIvLCPAQmXbwgODg0LCwoJCACGBQDQAG  
BAQIDAwQEBAUEAwQCAgICBAMFCAQFBgUGBwCICQgKGxwcHh8V/sMGBg8SExUXFfxkgIB8fHx4dHBs  
SF+IDBQoJCgsMDg4QEBESExQUFRUWHBkiHRkUDwsHAwEBbiAaGxwdHR4eCagIBwYGAGIBAgMFBsA  
fHh0bGhgfwAEXBi0dIh8aFg/+1yAZHq0LCQgHBQQDAGEDAwQJDhERExUXGBoc6QUJCQCgGfxMPDg0  
LCgkHBgUDAGEDBAMFbQYHCAGJDxAREhAQIyS5Dw4NDAsJCACFAwMDAQEBAGMGCAoLDRcaExh0BAM  
CAQICAwQJDA8RExUXFfxoaGhIkJCMhIR8ODg4MCwsQDwkHBgYE1wMDAGEBAGMEBAUFBgCHBwGICAK  
ICQgHCAcGBgYFBAMDAGICAgMFBQYGCAGKCgoMDBIJBwGKCwsNDQ4QDyMkJEMdHhwdHBwCGxoAGRg  
XFxYWFbQCTAEDBACICgsMDQ0PDxQAEbftBAUDAwMCAQEBAQIDAwMFBDCsIASWfgoKCgCHBgUFBAM  
CBQQGCQsIQoQoKExQTFBITERUSEA8NCwkGBQIF/ncEawQBAQEBAQIDBAUGBwgJCgsMERAZGRcWFR  
QCgcGvxowFBUTFBISAwEBAwUGBgYGBgYFBAQTFBQTFBUUG5gjAxkRGBgYFhIVFhUbDQ0LDAoKCgg  
HBgUFAwIEAgEBAwQGBwoLWwEFBwgH3BUSEREQEBAODw0NDAwKCgkGBQUEBAQDAwMCAgEBBAZBBQU  
HBwkJCgsICAgJCAkSExMTFBUFR8gFRnJCAgICQkJCQkJChMUFBQTFBMSHhEQChMQDQwIBgIBAQI  
EBgUFBgUGAXQTEYWNwgJCAgHBwYGBgUEAwMCAgICAwMEBQYGBgCHCAgJCG0RERAQEBApDw4ODQw  
LCgkKEwWNDQwNDQ0NDQ0ZGBUiDg4MCwsKCAgHBQUEAwEBAgMADAAAAAAD8gP0AAgADAAQABQAGAA  
cAEQASABMAFAAVABYAAATFSE1JwcnBycFMzcjNxc3JwcXNyc/ATUnBxUXNRcVHwg/CD0BLwcrAQ8  
HNxc3JwcXNyc7AScjJREhEQMhESF+AwSperIsRwFaCgYWRrWGFp8JHRCZiIiLOIhKDBAYICgoGBgc  
FDAoKCAyFAgEDBAYICQsGBgYGDaoKCAyFAgFxDxYGrBMPHEgWBgoBEPYURAPk/BwBr96cVT+yGUs  
DIhMWBxwcChYQLgcGBgYGCYJBgsLCQgHBQEBAQEBCACHCgsFBwYGCwsJCACFAQEBCQYICQsGBj8  
QHAYGHw8WI1H9BQL7/GMD6AAAAQAQAAAA/QDqAAGADYAPQBBAABNxmVITUBJRUFCTsBPwk9AS8  
KDwoLEQMHAwERAYERIJJg+v8kgEKAToBAQUHCAoGBQYHBgYGBgYGCQkHBAIBAQIEBwkJBgYGBgY  
GDAwYFBgoIBwQCAQEg7YL1/vY9A+j8GAFBqf7tQpYBR3oHBgYMCgkHAWICAQECAGMHCQoMBgYHBwY  
GBAoJBwMCAQEBAQEBAgMHCQoMBgZ5/cgBF6gBMP64AeH87ANQAAAAAAAEgDeAAAAAQA  
AAAAEAAAAAEACgABAAEAAAAAIABwLAEEAAAAAAMACgASAAEAAAAAQAACgAcAAEAAAAAAU  
ACwAmAAEAAAAAAYACgAxAAEAAAAAaOLAA7AAEAAAAAAsAEgBnAAMAAQQJAAAAAgB5AAMAAQQ  
JAAEFAB7AAMAAQQJAAIADgCPAAMAAQQJAAMAFACdAAMAAQQJAAQAFACxAMAAQQJAAUAFgDFAAM  
AAQQJAAYAFADbAAMAAQQJAAoAWADvAAMAAQQJAAsAJAFHIGZvbnQtaWNvbnNSZWdlbGZyZm9udC1  
pY29uc2ZvbnQtaWNvbnNSZWZJZaW9uIDEuMGZvbnQtaWNvbnNGb250IGdlbmVYXRlZCB1c2luZyB  
TeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAGYAbwBuAHQALQBPAGM  
AbwBuAHMAUGBlAGcAdQBsAGEAcgBmAG8AbgB0AC0AaQBJAG8AbgBzAGYAbwBuAHQALQBPAGMABwB  
uAHMAVGBlAHIAcWBPAG8AbgAgADEALgAwAGYAbwBuAHQALQBPAGMABwBuAHMARgBvAG4AdAAgAGC

```

AZQBuAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB
0AHIAbwAgAFMAdABlAGQAaQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwBtAAA
AAAIAAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACgECAQMBBAEFAQYBBwEIAQkBCgELAA
dZyXZlXzAyB3NhdUUtMDEHfZWRpdF8wMwdlZG10XzAxBNwNfZWZyDHBhaW50LWJlY2tldA9wYWludC1
idWNrZXQtd2YGaW1hZ2VzC3BpY3RlcmVzLXdmAAAA) format('truetype');
 font-weight: normal;
 font-style: normal;
}
.e-sign-icons {
 font-family: 'font-icons' !important;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
#signature-toolbar-control {
 border: 1px solid lightgray;
}
#signature-toolbar-control #toolbar {
 border: none;
 border-bottom: 1px solid lightgray;
 box-sizing: border-box;
}
#signature-toolbar-control #toolbar {
 height: 44px !important;
}
#signature-toolbar-control .e-btn:disabled {
 opacity: 0.5 !important;
 pointer-events: none;
}
#signature-toolbar-control #signature-control {
 height: 300px;
 width: 100%;
 margin: 0;
}
#signature-toolbar-control #signature {
 border: none !important;
 height: 90%;
 width: 100%;
}
.e-colorpicker-wrapper.e-bg-color #bg-color + .e-split-btn-wrapper .e-
split-btn .e-selected-color {
 background: none;
 border-bottom-style: solid;
 border-bottom-width: 3px;
 width: 14px;
 margin: 0px 2px;
 border-bottom-color: #ffffff;
}
.e-colorpicker-wrapper.e-stroke-color #stroke-color + .e-split-btn-
wrapper .e-split-btn .e-selected-color {
 background: none;
 border-bottom-style: solid;

```

```

 border-bottom-width: 3px;
 width: 14px;
 margin: 0px 2px;
 border-bottom-color: #000000;
 }
 .e-colorpicker-wrapper.e-bg-color #bg-color + .e-split-btn-wrapper
.e-split-btn .e-selected-color .e-split-preview,
 .e-colorpicker-wrapper.e-stroke-color #stroke-color + .e-split-btn-
wrapper .e-split-btn .e-selected-color .e-split-preview {
 display: none;
 }
 .e-colorpicker-wrapper.e-bg-color #bg-color + .e-split-btn-wrapper .e-
split-btn .e-selected-color::before {
 content: '\e707';
 }
 .e-colorpicker-wrapper.e-stroke-color #stroke-color + .e-split-btn-
wrapper .e-split-btn .e-selected-color::before {
 content: '\e704';
 }
 #signature-toolbar-control .e-clear::before {
 content: '\e706';
 }
 #signature-toolbar-control .e-save::before {
 content: '\e701';
 }
 /* Circle palette customization */
 .e-container .e-palette .e-circle-palette {
 border: 0;
 height: 32px;
 width: 32px;
 border-radius: 20px;
 margin: 4px;
 }
 .e-container .e-palette .e-circle-palette:hover {
 box-shadow: none;
 transform: scale(1.2);
 transition: transform .2s ease-out;
 }
 .e-circle-palette .e-circle-selection {
 height: 32px;
 width: 32px;
 border-radius: 20px;
 display: inline-block;
 transform: scale(0);
 transition: transform 1.2s ease-in;
 }
 .e-circle-palette.e-selected .e-circle-selection {
 transform: scale(0.8);
 background-color: #fff;
 transition: transform .2s ease-out;
 }
 #circle-palette + .e-container,
 #scroll-palette + .e-container {
 background-color: transparent;
 border-color: transparent;
 box-shadow: none;
 }
}

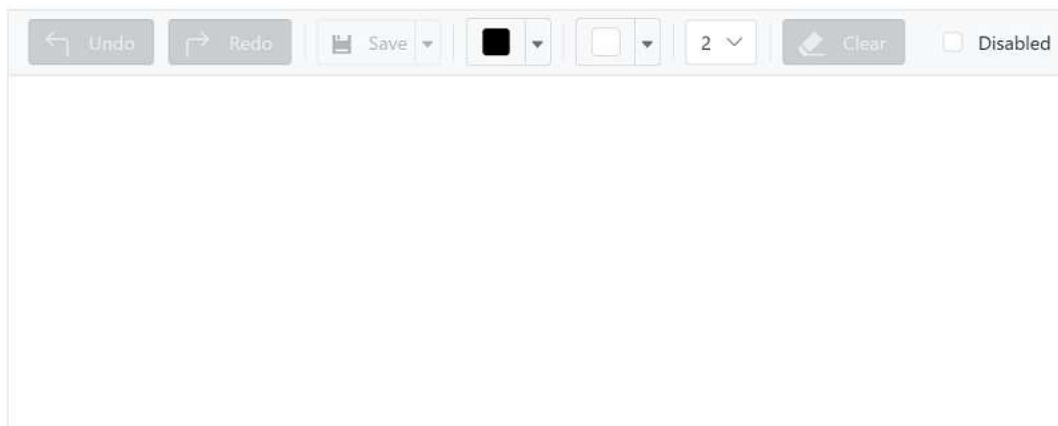
```

```
.e-container .e-palette .e-circle-palette.e-selected {
 outline: none;
}
.e-bg-color .e-circle-palette.e-nocolor-item.e-selected .e-circle-
selection {
 background: transparent;
}
.e-bg-color .e-circle-palette.e-nocolor-item.e-selected {
 border: 3px solid lightgray;
}
</style>
```

### DEFAULT.CS

```
public ActionResult Default()
{
 return View();
}
```

Output be like the below.



## Skeleton

### Getting Started with ASP.NET MVC Skeleton Control

This section briefly explains about how to include **ASP.NET MVC Skeleton** control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

## Add ASP.NET MVC Skeleton control

Now, add the Syncfusion ASP.NET MVC Skeleton control in `~/Views/Home/Index.cshtml` page.

**CSHTML**

```
@using Syncfusion.EJ2.Notifications;
<div class="col-sm-6">
<h5>Circle</h5>
@Html.EJS().Skeleton("skeletonCircleSmall").Shape(SkeletonType.Circle).Width
("3rem").Render()
@Html.EJS().Skeleton("skeletonCircleMedium").Shape(SkeletonType.Circle).Width
("48px").Render()
@Html.EJS().Skeleton("skeletonCircleLarge").Shape(SkeletonType.Circle).Width
("64px").Render()
@Html.EJS().Skeleton("skeletonCircleLarger").Shape(SkeletonType.Circle).Width
("80px").Render()
</div>
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Skeleton control will be rendered in the default web browser.



## Skeleton Types

The Skeleton control has the following different type of shapes.

- Circle
- Square
- Text
- Rectangle

**CSHTML**

```
@using Syncfusion.EJ2.Notifications;
<div class="row skeleton-default">
<div class="col-sm-6">
<h5>Circle</h5>
```

```

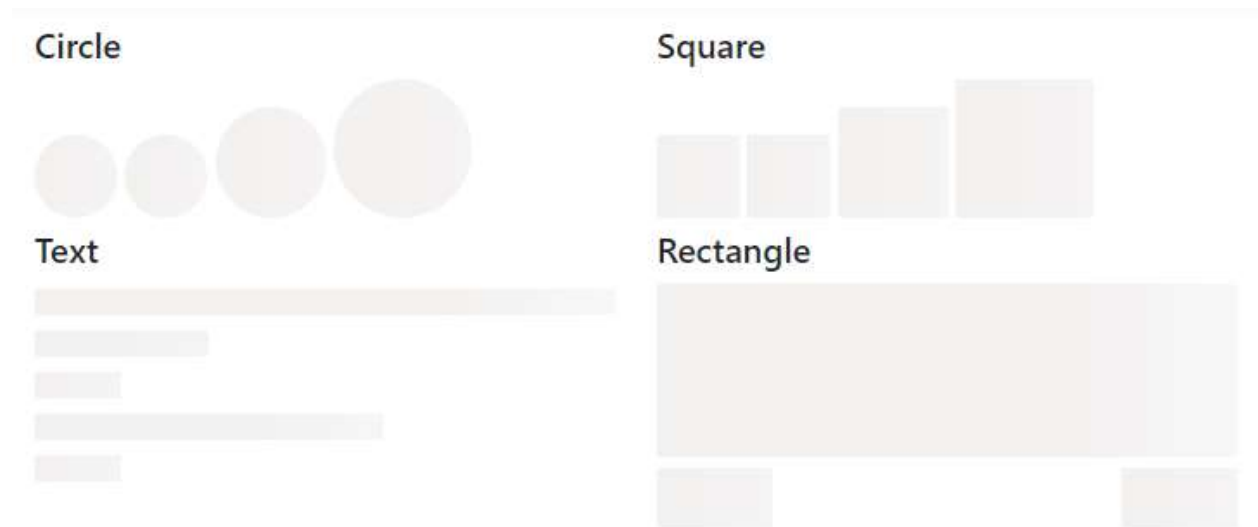
@Html.EJS().Skeleton("skeletonCircleSmall").Shape(SkeletonType.Circle).Width
("3rem").Render()
@Html.EJS().Skeleton("skeletonCircleMedium").Shape(SkeletonType.Circle).Width
("48px").Render()
@Html.EJS().Skeleton("skeletonCircleLarge").Shape(SkeletonType.Circle).Width
("64px").Render()
@Html.EJS().Skeleton("skeletonCircleLarger").Shape(SkeletonType.Circle).Width
("80px").Render()
</div>
<div class="col-sm-6">
<h5>Square</h5>
@Html.EJS().Skeleton("skeletonSquareSmall").Shape(SkeletonType.Square).Width
("3rem").Render()
@Html.EJS().Skeleton("skeletonSquareMedium").Shape(SkeletonType.Square).Width
("48px").Render()
@Html.EJS().Skeleton("skeletonSquareLarge").Shape(SkeletonType.Square).Width
("64px").Render()
@Html.EJS().Skeleton("skeletonSquareLarger").Shape(SkeletonType.Square).Width
("80px").Render()
</div>
</div>
<div class="row skeleton-default">
<div class="col-sm-6">
<h5>Text</h5>
@Html.EJS().Skeleton("skeletonText").Shape(SkeletonType.Text).Width("100%").
Height("15px").Render()
@Html.EJS().Skeleton("skeletonTextMedium").Width("30%").Height("15px").Rende
r()

@Html.EJS().Skeleton("skeletonTextSmall").Width("15%").Height("15px").Render
()

@Html.EJS().Skeleton("skeletonTextMedium1").Width("60%").Height("15px").Rend
er()

@Html.EJS().Skeleton("skeletonTextSmall1").Width("15%").Height("15px").Rende
r()
</div>
<div class="col-sm-6">
<h5>Rectangle</h5>
@Html.EJS().Skeleton("skeletonRectangle").Shape(SkeletonType.Rectangle).Width
("100%").Height("100px").Render()
@Html.EJS().Skeleton("skeletonRectangleMedium").Shape(SkeletonType.Rectangle)
.Width("20%").Height("35px").Render()
@Html.EJS().Skeleton("skeletonRectangleMediumRight").Shape(SkeletonType.Rect
angle).Width("20%").Height("35px").Render()
</div>
</div>

```



## Shapes in ASP.NET MVC Skeleton Control

The Skeleton control support various built-in shape variants to design layout of the page. You can use the [Shape](#) property to create a preview of any layout.

The Skeleton control supports the following content shapes:

### Circle skeleton shape

#### **CSHTML**

```
@using Syncfusion.EJ2.Notifications
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Circle).Width("48px").Render()
```

#### **CIRCLESHAPE.CS**

```
public ActionResult Circle()
{
 return View();
}
```



### Square skeleton shape

#### **CSHTML**

```
@using Syncfusion.EJ2.Notifications
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Square).Width("48px").Render()
```



**SQUARESHAPE.CS**

```
public ActionResult Square()
{
 return View();
}
```



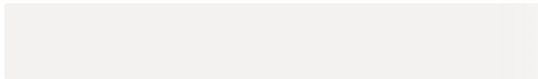
## Rectangle skeleton shape

**CSHTML**

```
@using Syncfusion.EJ2.Notifications
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Rectangle).Height("50px")
.Render()
```

**RECTANGLESHAPE.CS**

```
public ActionResult Rectangle()
{
 return View();
}
```



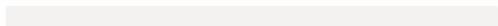
## Text skeleton shape

**CSHTML**

```
@using Syncfusion.EJ2.Notifications
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Text).Height("15px").Render()
```

**TEXTSHAPE.CS**

```
public ActionResult Text()
{
 return View();
}
```



Below example demonstrates the above functionalities of a Skeleton control.

### CSSHTML

```
@using Syncfusion.EJ2.Notifications
<div id="skeletonCard">
 <div class='cardProfile'>

@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Circle).Width("60px").Render()
 </div>
 <div class="cardinfo">

@Html.EJS().Skeleton("skeleton1").Width("30%").Height("15px").Render()

@Html.EJS().Skeleton("skeleton2").Width("15%").Height("15px").Render()
 </div>
 <div class="cardContent">

@Html.EJS().Skeleton("skeleton3").Shape(SkeletonType.Rectangle).Width("100%")
).Height("150px").Render()
 </div>
 <div class="cardoptions">

@Html.EJS().Skeleton("skeleton4").Shape(SkeletonType.Rectangle).Width("20%")
.Height("32px").Render()

@Html.EJS().Skeleton("skeleton5").Shape(SkeletonType.Rectangle).Width("20%")
.Height("32px").Render()
 </div>
</div>
<style>
 #skeletonCard {
 padding: 10px;
 line-height: inherit;
 height: 330px;
 }
 #skeletonCard .cardProfile {
 float: left;
 margin-right: 15px;
 }
 #skeletonCard .cardinfo {
 margin-top: 10px;
 overflow: hidden;
 }
 #skeletonCard .cardContent {
 margin: 20px 0px 20px;
 }
 #skeletonCard .cardoptions {
 display: flex;
 justify-content: space-between;
 }
</style>
```

### SHAPE.CS

```
public ActionResult Shape()
```

```
{
 return View();
}
```



### Shimmer Effect in ASP.NET MVC Skeleton Control

You can use the [ShimmerEffect](#) property to change animation effect in the skeleton control. Skeleton supports Wave, Pulse and Fade effects and by default, the ShimmerEffect is set to Wave effect.

#### CSHTML

```
@using Syncfusion.EJ2.Notifications
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Circle).Width("60px").ShimmerEffect(ShimmerEffect.Pulse).Render()
```

#### PULSEEFFECT.CS

```
public ActionResult PulseEffect()
{
 return View();
}
```



Below example demonstrates a list with pulse effect skeleton.

#### CSHTML

```
@using Syncfusion.EJ2.Notifications
```

```

<ul id="skeleton-list" class="e-card">

 <div class='cardProfile'>
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Circle).Width("40px").ShimmerEffect(ShimmerEffect.Pulse).Render()
 </div>
 <div>
@Html.EJS().Skeleton("skeleton1").Width("60%").Height("15px").ShimmerEffect(ShimmerEffect.Pulse).Render()

@Html.EJS().Skeleton("skeleton2").Width("40%").Height("15px").ShimmerEffect(ShimmerEffect.Pulse).Render()
 </div>

 <div class='cardProfile'>
@Html.EJS().Skeleton("skeleton3").Shape(SkeletonType.Circle).Width("40px").ShimmerEffect(ShimmerEffect.Pulse).Render()
 </div>
 <div>
@Html.EJS().Skeleton("skeleton4").Width("60%").Height("15px").ShimmerEffect(ShimmerEffect.Pulse).Render()

@Html.EJS().Skeleton("skeleton5").Width("40%").Height("15px").ShimmerEffect(ShimmerEffect.Pulse).Render()
 </div>

<style>
 #skeleton-list {
 padding-left: 12px;
 padding-top: 7px;
 line-height: inherit;
 }
 #skeleton-list li {
 list-style: none;
 display: flow-root;
 margin-bottom: 9px;
 }
 .cardProfile {
 float: left;
 margin-right: 15px;
 }
</style>

```

**EFFECT.CS**

```

public ActionResult Effect()
{
 return View();
}

```



## Styles in ASP.NET MVC Skeleton Control

You can customize skeleton control in the below ways.

### CssClass

You can customize the style of a Skeleton control by using [CssClass](#). The appearance of ASP.NET MVC Skeleton can be customized by changing the wave color, background color, width, and height.

### CSHTML

```
@using Syncfusion.EJ2.Notifications
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Circle).Width("60px").CssClass("e-customize").Render()

<style>
 .e-customize.e-skeleton.e-shimmer-wave::after {
 background-image: linear-gradient(90deg, transparent calc(50% - 100px), rgb(30 128 234 / 50%) 50%, transparent calc(50% + 100px));
 }
 .e-customize.e-skeleton.e-skeleton-circle {
 background-color: #a8c1f2;
 }
</style>
```

### CUSTOMIZE.CS

```
public ActionResult Customize()
{
 return View();
}
```



### Visible

You can use the [Visible](#) property which defines the visible state of Skeleton.

### CSHTML

```
@using Syncfusion.EJ2.Notifications
```

```
@Html.EJS().Skeleton("skeleton").Shape(SkeletonType.Circle).Width("60px").Visible(false).Render()
```

## VISIBLE.CS

```
public ActionResult Visible()
{
 return View();
}
```

## Accessibility in ASP.NET MVC Skeleton Control

Accessibility is achieved in the Skeleton control through WAI-ARIA standard. The Skeleton control can be effectively accessed through assistive technologies such as screen readers.

### ARIA attributes

The Skeleton control characterized with complete ARIA accessibility support that helps to be accessible by on-screen readers and other assistive technology devices. The following ARIA attributes are applicable for Skeleton control.

| Properties | Functionality                                                                     |
|------------|-----------------------------------------------------------------------------------|
| role       | This attribute is added to the input element to describe the actual role.         |
| aria-label | Attribute provides the text label with some default description for the Skeleton. |
| aria-live  | The aria-live attribute indicates the priority of updates to a live region.       |
| aria-busy  | This attribute is set to true when component is shown.                            |

## Smith Chart

### Getting Started with ASP.NET MVC SmithChart Control

This section briefly explains about how to include [ASP.NET MVC SmithChart](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

#### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

### Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

## PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

##### ~/ \_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

##### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

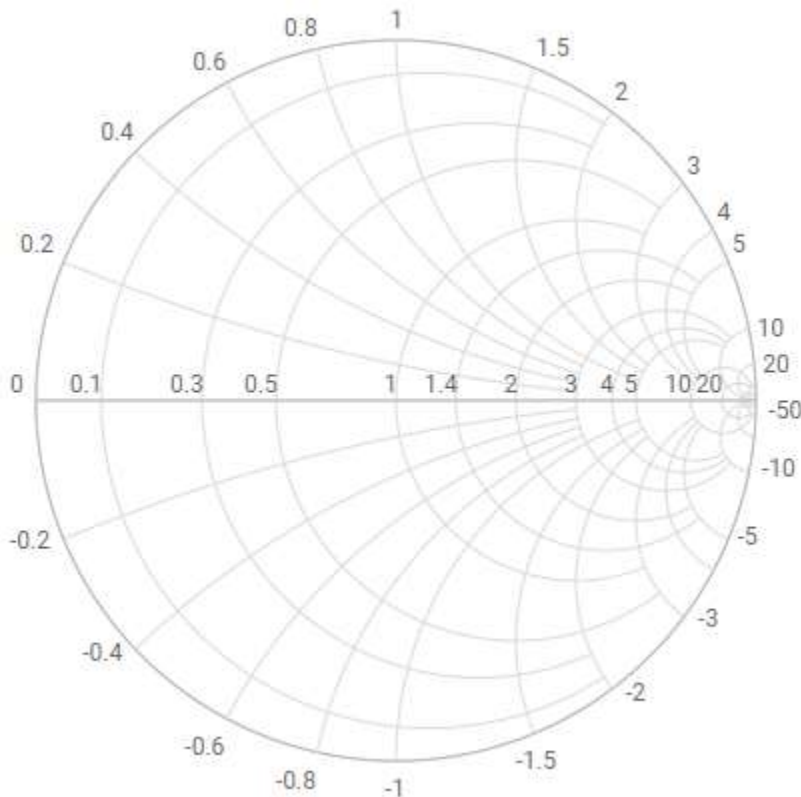
#### Add ASP.NET MVC SmithChart control

Now, add the Syncfusion ASP.NET MVC SmithChart control in **~/Views/Home/Index.cshtml** page.

##### CSHTML

```
@Html.EJS().Smithchart("smith").Render();
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC SmithChart control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

### Working with Data

Smithchart can visualise the data bound from local data. The data you bind for the smithchart, should be an array of object and that should contain the field resistance and reactance. This should be bind to points or datasource in the smithchart.

### Data Binding

You can bind simple JSON data to smithchart using point property in series. JSON data should contain [resistance] and [reactance] fields. This JSON data should be bind to points or datasource in the smithchart. You can any number of JSON for points or datasource as per your requirement.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

<script>
function loaded(args) {
 window.smithchart = args.smithchart;
```



```

 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 }
];
 args.smithchart.series[1].points = [{ resistance: 0, reactance:
0.15 }, { resistance: 0, reactance: 0.15 },
 { resistance: 20, reactance: -50 }, { resistance: 10,
reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

## WORKING-WITH-DATA.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

```
}
```

## Smithchart Dimensions

You can render the smithchart either corresponding to its container size or you can set the size of the smithchart as per your requirement. To render the smithchart corresponding to its container size, you need to set the size for the smithchart container. Else to set the size for the smithchart as per your requirement, you can use the width and height properties in the smithchart.

### Size for Container

You can render smithchart to it's container size. To achieve this, you need to specify the width and height of the smithchart's container via inline or CSS as demonstrated below.

```
`javascript
<div id='container'>
<div id='element' style="width:650px; height:350px;"></div>
</div>
`
```

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
}).Render()

<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>
```

### CONTAINER.CS

```
using System;
using System.Collections.Generic;
```

```
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```

### Size for Smithchart

<!-- markdownlint-disable MD036 -->

You can also set size for smithchart directly through [width] and [height] properties. Using this properties, you can directly mention the width and height of the smithchart in pixels or you can set the width and height in percentage.

#### In Pixel

In smithchart's width and height property, you can directly give values in pixels like below demonstration. This will render smithchart in same size as you mentioned in you code.

#### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Height("300px").Width(
"650px").Series(series =>
{
 series.Name("Transmission1").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
}
```

```

 args.smithchart.refresh();
 }
</script>

```

### SIZE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### In percentage

You can also specify the width and height of the smithchart in percentage. If you mention the width and height in percentage, then smithchart will be render as per the percentage of it's container size. You can set the values in percentage like below demonstration.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Height("90%").Width("85%").Series(series =>
{
 series.Name("Transmission1").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
]
}

```

```

];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

## PERCENTAGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

## Title and Subtitle

### Enable title

Title and subtitle is used to depicts the information about the data plotted in the smithchart. You can set the title and subtitle of the smithchart using the [text] property in title and subtitle. By default visibility of the title as well as subtitle is enabled. You need to set simply text for title and subtitle in your sample as like below.

## CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Title(title =>
title.Text("Transmission details").Visible(true).Subtitle(subtitle
=>subtitle.Text('Transmission')).Series(series =>
{
 series.Name("Transmission1").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },

```

```

 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### TITLE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Title trim

Both title and subtitle of the smithchart can be trimmed if it exceeds the certain length. Trimming is enabled using [enableTrim] for title as well as subtitle. This length can be changed using the property [maximumWidth]. Also [font], [textAlignment] and [visibility] can be customized for title as well as subtitle.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Title(title =>
title.Text("Transmission
details")).Visible(true).EnableTrim(true).Subtitle(subtitle
=>subtitle.Text('Transmission')).Series(series =>
{
 series.Name("Transmission1").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [

```

```

 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### TITLE-TRIM.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Axis in Smithchart Control

Like chart, Smithchart is having support for two types of axis.

- Horizontal axis - axis drawn as straight line in the horizontal direction of the chart.
- Radial axis - axis is drawn as circular path.

### Labels Customization

Axis labels are used to denote what kind of data is bound for smithchart. Using axis labels, you can easily identify in which interval chart is rendered. Using following properties we can customize the axis labels for horizontal and radial axis.

- `[labelPosition]` - used to place the labels either inside or outside the axis line.
- `[labelIntersectAction]` - used to hide the labels when intersect with other one.

- `[labelStyle]` - used to customize the properties such as font size, family, weight, opacity.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.horizontalAxis.majorGridLines.visible = true;
 args.smithchart.horizontalAxis.majorGridLines.opacity = 0.8;
 args.smithchart.horizontalAxis.majorGridLines.width = 10;
 args.smithchart.horizontalAxis.axisLine.visible = true;
 args.smithchart.horizontalAxis.axisLine.width = 10;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}

</script>
```

### LABEL.CS



```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Gridlines

To make the data in a chart that displays axes easier to read, you can display horizontal and radial axis gridlines. Gridlines extend from any horizontal and radial axes across the plot area of the smithchart.

Both horizontal and radial axis are having support for major as well as minor gridlines. Major gridlines are drawn from the position in which labels are rendered. Minor gridlines are drawn between two major gridlines as per the count we set in settings.

We can customize following things, in major as well as minor gridlines.

- [width] - used to customize the width of gridlines.
- [dashArray] - used to customize whether gridline have to render as normal line or dashed line.
- [visible] - used to enable or disable the visibility of the gridlines.
- [opacity] - used to customize the opacity of the major gridlines.
- [count] - used to customize the count of the minor gridlines.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.horizontalAxis.majorGridLines.visible = true;
 args.smithchart.horizontalAxis.majorGridLines.opacity = 0.8;
 args.smithchart.horizontalAxis.majorGridLines.width = 10;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },

```

```

 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### GRID-LINE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Axisline

As name suggests that, it is a line in smithchart that can be configured to denotes the axis. By default, visibility of the axis line is true. You can customize its visibility by using visible property in axis Line. Other than visibility of the axis line, you can customize the following properties of the axis line.

- **[width]** - used to customize the width of the axis line.
- **[dashArray]** - used to render the axis line as dashed line.
- **[visible]** - used to enable or disable the visibility of the axis line.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.horizontalAxis.labelPosition = 'Inside';
 args.smithchart.horizontalAxis.labelIntersectAction = 'Hide';
 args.smithchart.radialAxis.labelPosition = 'Inside';
 args.smithchart.radialAxis.labelIntersectAction = 'Hide';
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },];
}
```

```

 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

### AXIS-LINE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

<!-- markdownlint-disable MD036 -->

### Legend

Legend is a key used in smithchart, that contains symbol and descriptions. It provides valuable information for interpreting what the smithchart is displaying and can be represented in various colors, shapes or other identifiers based on the data. In simple words, we can define that legend is used to denote the series rendered in the smithchart.

### Position and Alignment

By default visibility of the legend is false. To enable the legend, set visible as true in legendSettings. Default position for the legend is bottom. By using [position] property, you can change the position of the legend. You can either place the legend at bottom, top, right and left side of the smithchart. To use the legend in smithchart, you need to import and inject the SmithchartLegend from chart.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).Position('Top'))Series(series =>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()
<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [

```

```

 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

## POSITION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

Other than these positions, you can place the legend anywhere in the smithchart. To achieve this, you have to set position as custom in legendSettings and specify the x and y coordinates using the x and y properties in the location.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).Position('Top').Location =>(new
SmithchartLocation{x:80,y:90}))Series(series =>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>
```

### CUSTOM-POSITION.CS

```
using System;
```

```

using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Legend Alignment

Other than positioning the legend in the smithchart, you can customize its alignment also. By default, legend is aligned at center. Using the [alignment] property, you can align the legend in near and far locations of the smithchart.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).Position('Top').Alignment('Near'))Series(series =>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },

```

```

 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

### ALIGNMENT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Customization

#### Legend Shape

By default, legend is rendered in the circle shape and the color of the shape is as same as series color in the smithchart. Using the property [shape] in legend settings, you can change the icon shape of the legend as rectangle, triangle and so on.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).Position('Top').Shape('Rectangle'))Series(series =>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()

```



```

<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

### CUSTOM-SHAPE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

```

 }
}

```

### Legend Size

By default, legend takes 20% - 25% of the chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the width vertically, while placing on left or right position of the chart. You can change this default legend size by using the `[width]` and `[height]` property of the `legendSettings`.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).Position('Top').ItemPadding(5).ShapePadding(10))Series
(series =>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}

```

```
</script>
```

### LEGEND-SIZE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```

### Padding

You can customize the space between two legend items and space between legend shape and text as per your requirement. For customizing the space between two legend items, you can use `[itemPadding]` property. To control space between legend shape and text, you can use `[shapePadding]` property.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).position('Top').Height(100).Width(200))Series(series
=>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
```

```

 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### PADDING.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Toggle Visibility

By default series name is displayed in the legend. You can collapse the visibility of the series by clicking the legend for the particular series. You can toggle the series visibility as true or false using the [toggleVisibility] property. By default it is true.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;

```

```

@Html.EJS().Smithchart("smithchart").Loaded("loaded").LegendSettings(legend=
>legend.Visible(true).Position('Top').toggleVisibility(true))Series(series
=>
 {
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
 }).Render()
<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 20, reactance:
-50 }, { resistance: 10, reactance: -10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

**TOGGLE.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers

```

```
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```

## Tooltip

Smithchart will display details about the points through tooltip, when the mouse is moved over the point. By default, tooltip is disabled. To enable the tooltip for smithchart, you need to import and inject TooltipRender module from chart. And also set the property visible as true, in tooltip settings. You can customize the tooltip's visibility and appearance differently each series in the smithchart.

## CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Tooltip(new { visible = true
}).Marker(marker => marker.Visible(true)).Add();
 series.Name("Transmission2").Tooltip(new { visible = true
}).Marker(marker => marker.Visible(true)).Add();
}).Render()
<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.series[1].points = [{ resistance: 0, reactance:
0.15 }, { resistance: 0, reactance: 0.15 },
 { resistance: 20, reactance: -50 }, { resistance: 10, reactance:
-10 },
 { resistance: 9, reactance: -4.5 }, { resistance: 8,
reactance: -3.5 },
 { resistance: 7, reactance: -2.5 }, { resistance: 6,
reactance: -1.5 },
 { resistance: 5, reactance: -1 }, { resistance: 4.5,
reactance: -0.5 },
```

```

 { resistance: 3.5, reactance: 0 }, { resistance: 2.5,
reactance: 0.4 },
 { resistance: 2, reactance: 0.5 }, { resistance: 1.5,
reactance: 0.5 },
 { resistance: 1, reactance: 0.4 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0,
reactance: 0.05 },,];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

### TOOLTIP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

<!-- markdownlint-disable MD036 -->

### Marker & Datalabels

Markers and Datalabels are used to provide information about the data points in the series. You can add a shape to adorn each data point. By default marker and datalabel both are disabled in smithchart. You can enable both of them by setting visible property as true in marker and datalabel settings

#### Marker

Default visibility of marker is false. You can enable the marker by setting property visible as true in marker settings. This will add marker for each point in the series. Using marker setting, you can customize marker differently for each series in smithchart.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

```

```

<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].marker = {
 shape: 'Circle',
 visible: true,
 border: {
 width: 2,
 }
 };
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

### MARKER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Marker Customization

Using marker settings in series, you can customize the marker for each series differently. Using marker settings, you can customize following properties differently for each series in the smithchart.



- [width] - To control the width of the marker.
- [height] - To control the height of the marker.
- [fill] - Used to customize the fill color of the marker.
- [opacity] - Used to customize the opacity of the marker.
- [border] - Used to control the width and color of the marker's border.
- [shape] - Used to change the shape of the marker.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].marker = {
 visible: true,
 height: 10,
 width: 10,
 fill: '#ff99ff',
 opacity: 1,
 shape: 'rectangle',
 border: {
 color: '#cc00cc',
 width: 2
 }
 };
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}

</script>
```

### CUSOM-MARKER.CS

```
using System;
using System.Collections.Generic;
```

```

using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Datalabels

By default, datalabel is disabled. You can enable the datalabel by setting property visible as true in datalabel settings. For each point in series, data label is created. Datalabel for each series can be customized differently using datalabel settings.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].marker = {
 dataLabel: {
 visible: true,
 fill: '#99ffcc',
 opacity: 1,
 border: {
 color: '#1aff8c',
 width: 2,
 }
 }
 };
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },

```

```

 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### DATA-LABEL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Datalabel customization

Using datalabel settings in marker, you can customize the datalabel for each series differently. In datalabel, you can customize the following properties differently for each series.

- **[fill]** - Used to changes the fill color of the data label's shape.
- **[opacity]** - Used to control the opacity of the data label's shape.
- **[border]** - Used to customize the width and color of the border.
- **[textStyle]** - Used to customize the font color, width and size.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].marker = {
 dataLabel: {
 visible: true,

```

```

 }
 };
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### CUSTOM-LABEL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Series

You can add any number of series to the smithchart as per your requirement. You can use series setting to either add or customize the data. For the points or datasource added in the series, line is drawn. You can customize the each series as per your requirement with marker, datalabel, animation, opacity and so on.

#### points or datasource

For adding values in the smithchart, you can use either points or datasource in the series. Points and datasource both should be array of object which should contain the field names resistance and reactance.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{

series.Name("Transmission1").Fill('#009933').Visibility('visible').Opacity('
0.75').Width(2.5).Marker(marker => marker.Visible(true).DataLabel(dataLabel
=> dataLabel.Visible(true))).Add();
}).Render()

<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
</script>

```

### SERIES.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Series customization

Using following options in series settings, you can customize each series in smithchart as per your requirement.

- [fill] - Used to customize the fill color for the series.
- [enableSmartLabels] - Used to place the data labels on the smithchart without overlapping with each other.
- [visibility] - Used to handle the visibility of the series.
- [opacity] - Used to control the opacity of the series line.
- [width] - Used to customize the width of the series line.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
 { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
 { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
 { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
 { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
];
 args.smithchart.series[1].points = [{ resistance: 0, reactance:
0.15 }, { resistance: 0, reactance: 0.15 },
 { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
 { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
 { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
```

```

 { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
 { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
 { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
 }];

 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
</script>

```

### CUSTOM-SERIES.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

## Print and Export

### Print

The rendered smithchart can be printed directly from the browser by calling the public method print. ID of the smithchart's div element must be passed as argument to that method.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Print").CssClass("e-flat").Render()
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
}).Render()
<script>
function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 10, reactance: 25 }, { resistance: 8,
reactance: 6 },
 { resistance: 6, reactance: 4.5 }, { resistance: 4.5,
reactance: 2 },

```

```

 { resistance: 3.5, reactance: 1.6 }, { resistance: 2.5,
reactance: 1.3 },
 { resistance: 2, reactance: 1.2 }, { resistance: 1.5,
reactance: 1 },
 { resistance: 1, reactance: 0.8 }, { resistance: 0.5,
reactance: 0.4 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0,
reactance: 0.15 },
];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
}
document.getElementById('togglebtn').onclick = function () {
 var smithchart =
document.getElementById('smithchart').ej2_instances[0];
 smithchart.print();
};
</script>

```

### PRINT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Export

The rendered smithchart can be exported to JPEG , PNG, SVG or PDF format by using export method in smithchart. Input parameters for this method are Export Type for format and fileName of result.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Charts;
@Html.EJS().Button("togglebtn").IconCss("e-icons e-play-
icon").Content("Print").CssClass("e-flat").Render()
@Html.EJS().Smithchart("smithchart").Loaded("loaded").Series(series =>
{
 series.Name("Transmission1").Add();
 series.Name("Transmission2").Add();
}).Render()

```



```

<script>
 function loaded(args) {
 window.smithchart = args.smithchart;
 args.smithchart.series[0].points = [
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0, reactance: 0.05 }, { resistance: 0,
reactance: 0.05 },
 { resistance: 0.3, reactance: 0.1 }, { resistance: 0.5,
reactance: 0.2 },
 { resistance: 1.5, reactance: 0.5 }, { resistance: 2.0,
reactance: 0.5 },
 { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
 { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
 { resistance: 2.5, reactance: 0.4 }, { resistance: 3.5,
reactance: 0.0 },
 { resistance: 4.5, reactance: -0.5 }, { resistance: 5.0,
reactance: -1.0 }
];
 args.smithchart.series[1].points = [{ resistance: 0, reactance:
0.15 }, { resistance: 0, reactance: 0.15 },
 { resistance: 0, reactance: 0.15 }, { resistance: 0.3,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
 { resistance: 0.3, reactance: 0.2 }, { resistance: 0.3,
reactance: 0.2 },
 { resistance: 0.5, reactance: 0.4 }, { resistance: 1.0,
reactance: 0.8 },
 { resistance: 2.5, reactance: 1.3 }, { resistance: 3.5,
reactance: 1.6 },
 { resistance: 3.5, reactance: 1.6 }, { resistance: 3.5,
reactance: 1.6 },
 { resistance: 4.5, reactance: 2.0 }, { resistance: 6.0,
reactance: 4.5 },
 { resistance: 8, reactance: 6 }, { resistance: 10, reactance: 25
 }];
 args.smithchart.loaded = null;
 args.smithchart.refresh();
 }
 document.getElementById('togglebtn').onclick = function () {
 var smithchart =
document.getElementById('smithchart').ej2_instances[0];
 smithchart.export('PNG', 'Smithchart');
 }
</script>

```

**EXPORT.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Accessibility in ASP.NET MVC Smith chart component

The Smith chart component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Smith chart component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

### WAI-ARIA attributes

The Smith chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Smith chart component:

- `img (role)`
- `region (role)`
- `aria-label (attribute)`
- `aria-hidden (attribute)`

### Keyboard interaction

The Smith chart component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Smith chart component.

| **Press** | **To do this** |

| --- | --- |

| **Tab** | Moves the focus to the next element in the Smith chart. |

| **Shift + Tab** | Moves the focus to the previous element in the Smith chart. |

| **Ctrl + P** | Prints the Smith chart. |

### Ensuring accessibility

The Smith chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Smith chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Smith chart component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## Sparkline

### Getting Started with ASP.NET MVC Sparkline Control

This section briefly explains about how to include [ASP.NET MVC Sparkline](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add script resources

Here, the script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### **~/\_LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
```

```
</head>
```

**Note:** Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC Sparkline control

Now, add the Syncfusion ASP.NET MVC Sparkline control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
<h2> Essential JS 2 for ASP.NET MVC Sparkline </h2>
@Html.EJS().Sparkline("spark").DataSource(Model).XName("xval").YName("yval")
.Height("100").Width("70").Render()
```

#### HOMECONTROLLER.CS

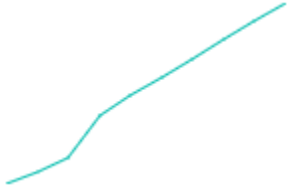
```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(DataSource.GetData());
 }
}

public class DataSource
{
 public int x;
 public string xval;
 public double yval;
 public static List<DataSource> GetData()
 {
 List<DataSource> data1 = new List<DataSource>();
 data1.Add(new DataSource() { x = 0, xval = "2005", yval = 20090440 });
 data1.Add(new DataSource() { x = 1, xval = "2006", yval = 20264080 });
 data1.Add(new DataSource() { x = 2, xval = "2007", yval = 20434180 });
 data1.Add(new DataSource() { x = 3, xval = "2008", yval = 21007310 });
 data1.Add(new DataSource() { x = 4, xval = "2009", yval = 21262640 });
 data1.Add(new DataSource() { x = 5, xval = "2010", yval = 21515750 });
 data1.Add(new DataSource() { x = 6, xval = "2011", yval = 21766710 });
 data1.Add(new DataSource() { x = 7, xval = "2012", yval = 22015580 });
 data1.Add(new DataSource() { x = 8, xval = "2013", yval = 22262500 });
 data1.Add(new DataSource() { x = 9, xval = "2014", yval = 22507620 });
 return data1;
 }
}
```

```
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Sparkline control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

## Sparkline Dimensions

Size for container

Sparkline can be rendered to its container size. You can set the size through inline or CSS as shown in the following code.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("350px").Width("650px").XName("xval").YName("yval").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

### SIZE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
```

```

public class HomeController : Controller
{
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval =
20090440 });
 data2.Add(new DataSource() { x = 1, xval = "2006", yval =
20264080 });
 data2.Add(new DataSource() { x = 2, xval = "2007", yval =
20434180 });
 data2.Add(new DataSource() { x = 3, xval = "2008", yval =
21007310 });
 data2.Add(new DataSource() { x = 4, xval = "2009", yval =
21262640 });
 data2.Add(new DataSource() { x = 5, xval = "2010", yval =
21515750 });
 data2.Add(new DataSource() { x = 6, xval = "2011", yval =
21766710 });
 data2.Add(new DataSource() { x = 7, xval = "2012", yval =
22015580 });
 data2.Add(new DataSource() { x = 8, xval = "2013", yval =
22262500 });
 data2.Add(new DataSource() { x = 9, xval = "2014", yval =
22507620 });
 return data2;
 }
 }
}

```

<!-- markdownlint-disable MD036 -->

Size for sparkline

<!-- markdownlint-disable MD036 -->

You can also set the size for sparkline directly using the [width](#) and [height](#) properties.

### In pixel

You can set the size for sparkline in pixel as demonstrated in the following code.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">

```

```

@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150").Width("350")
.XName("xval").YName("yval").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

**PIXEL.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval =
20090440 });
 data2.Add(new DataSource() { x = 1, xval = "2006", yval =
20264080 });
 data2.Add(new DataSource() { x = 2, xval = "2007", yval =
20434180 });
 data2.Add(new DataSource() { x = 3, xval = "2008", yval =
21007310 });
 }
 }
 }
}

```



```

21262640 });
data2.Add(new DataSource() { x = 4, xval = "2009", yval =
21515750 });
data2.Add(new DataSource() { x = 6, xval = "2011", yval =
21766710 });
data2.Add(new DataSource() { x = 7, xval = "2012", yval =
22015580 });
data2.Add(new DataSource() { x = 8, xval = "2013", yval =
22262500 });
data2.Add(new DataSource() { x = 9, xval = "2014", yval =
22507620 });
return data2;
 }
}
}

```

### In percentage

By setting values in percentage, sparkline gets its dimension with respect to its container. For example, when the height is set to '50%', sparkline is rendered to half of its container height.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("50%").Width("350%")
.XName("xval").YName("yval").DataSource(ViewBag.sparkData).Render()
</div>
<style>
.spark {
border: 1px solid rgb(209, 209, 209);
border-radius: 2px;
width: 100%;
height: 100%;
}
</style>
<script>
function loaded(args)
{
window.sparkline = args.sparkline;
args.sparkline.loaded = null;
args.sparkline.refresh();
}
</script>

```

### PERCENTAGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;

```

```

using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval =
20090440 });
 data2.Add(new DataSource() { x = 1, xval = "2006", yval =
20264080 });
 data2.Add(new DataSource() { x = 2, xval = "2007", yval =
20434180 });
 data2.Add(new DataSource() { x = 3, xval = "2008", yval =
21007310 });
 data2.Add(new DataSource() { x = 4, xval = "2009", yval =
21262640 });
 data2.Add(new DataSource() { x = 5, xval = "2010", yval =
21515750 });
 data2.Add(new DataSource() { x = 6, xval = "2011", yval =
21766710 });
 data2.Add(new DataSource() { x = 7, xval = "2012", yval =
22015580 });
 data2.Add(new DataSource() { x = 8, xval = "2013", yval =
22262500 });
 data2.Add(new DataSource() { x = 9, xval = "2014", yval =
22507620 });
 return data2;
 }
 }
 }
}

```

## Sparkline Types

Different types of shapes can be used to represent the sparkline. You can change the sparkline type by setting the type property. Sparkline supports the following types:

- Line
- Column
- Win-Loss
- Pie
- Area

The following code sample shows different types of sparklines.

<!-- markdownlint-disable MD036 -->

## Line

The [Line](#) type is used to render the sparkline series as line.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("100px").Width("70%").XName("xval").YName("yval").Type(SparklineType.Line).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

### LINE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
```

```

 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval =
20090440 });
 data2.Add(new DataSource() { x = 1, xval = "2006", yval =
20264080 });
 data2.Add(new DataSource() { x = 2, xval = "2007", yval =
20434180 });
 data2.Add(new DataSource() { x = 3, xval = "2008", yval =
21007310 });
 data2.Add(new DataSource() { x = 4, xval = "2009", yval =
21262640 });
 data2.Add(new DataSource() { x = 5, xval = "2010", yval =
21515750 });
 data2.Add(new DataSource() { x = 6, xval = "2011", yval =
21766710 });
 data2.Add(new DataSource() { x = 7, xval = "2012", yval =
22015580 });
 data2.Add(new DataSource() { x = 8, xval = "2013", yval =
22262500 });
 data2.Add(new DataSource() { x = 9, xval = "2014", yval =
22507620 });
 return data2;
 }
 }
}

```

## Column

The [Column](#) type is used to render the sparkline series as column.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("200px").XName("xval").YName("yval").Type(SparklineType.Line).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

**COLUMN.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval =
20090440 });
 data2.Add(new DataSource() { x = 1, xval = "2006", yval =
20264080 });
 data2.Add(new DataSource() { x = 2, xval = "2007", yval =
20434180 });
 data2.Add(new DataSource() { x = 3, xval = "2008", yval =
21007310 });
 data2.Add(new DataSource() { x = 4, xval = "2009", yval =
21262640 });
 data2.Add(new DataSource() { x = 5, xval = "2010", yval =
21515750 });
 data2.Add(new DataSource() { x = 6, xval = "2011", yval =
21766710 });
 data2.Add(new DataSource() { x = 7, xval = "2012", yval =
22015580 });
 data2.Add(new DataSource() { x = 8, xval = "2013", yval =
22262500 });
 data2.Add(new DataSource() { x = 9, xval = "2014", yval =
22507620 });
 return data2;
 }
 }
 }
}

```

## Pie

The [Pie](#) type is used to render the sparkline series as pie.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("70%").XName("xval").YName("yval").Type(SparklineType.Pie).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

### PIE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
```

```

20090440 });
20264080 });
20434180 });
21007310 });
21262640 });
21515750 });
21766710 });
22015580 });
22262500 });
22507620 });

 return data2;
 }
}
}
}

```

## Win Loss

The [WinLoss](#) type is used to render the sparkline series as Win Loss.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("100px").Width("70%").XName("xval").YName("yval").Type(SparklineType.WinLoss).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### WINLOSS.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval = -
20090440 });
 data2.Add(new DataSource() { x = 1, xval = "2006", yval = -
20264080 });
 data2.Add(new DataSource() { x = 2, xval = "2007", yval = -
20434180 });
 data2.Add(new DataSource() { x = 3, xval = "2008", yval = -
21007310 });
 data2.Add(new DataSource() { x = 4, xval = "2009", yval = -
21262640 });
 data2.Add(new DataSource() { x = 5, xval = "2010", yval = -
21515750 });
 data2.Add(new DataSource() { x = 6, xval = "2011", yval = -
21766710 });
 data2.Add(new DataSource() { x = 7, xval = "2012", yval = -
22015580 });
 data2.Add(new DataSource() { x = 8, xval = "2013", yval = -
22262500 });
 data2.Add(new DataSource() { x = 9, xval = "2014", yval = -
22507620 });
 return data2;
 }
 }
 }
}

```

## Area

The [Area](#) type is used to render the sparkline series as area.



**CSHTML**

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("100px").Width("70%").XName("xval").YName("yval").Type(SparklineType.Area).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

**AREA.CS**

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "2005", yval = 20090440 });
 }
 }
 }
}
```

```

20264080 });
20434180 });
21007310 });
21262640 });
21515750 });
21766710 });
22015580 });
22262500 });
22507620 });
 return data2;
 }
}
}

```

## Axis Customization

You can customize axis value types and min and max values of the sparkline.

### Change value type of the sparkline

You can change the sparkline value type by setting the [valueType](#) property to **Numeric**, **Category**, or **DateTime**.

```
<!-- markdownlint-disable MD036 -->
```

### DateTime

You can assign date-time values to the sparkline by setting the [valueType](#) property to **DateTime**.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">

@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("130px").DataSource(ViewBag.datetime).Type(SparklineType.Column).XName("xDate").YName("yval").ValueType(SparklineValueType.DateTime).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)

```

```

 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

## DATETIME.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.datetime = DataSource.GetDatetimeData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetDatetimeData()
 {
 List<DataSource> data1 = new List<DataSource>();
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
1), yval = 4 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
2), yval = 4.5 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
3), yval = 8 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
4), yval = 7 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
5), yval = 6 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
8), yval = 8 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
9), yval = 8 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
10), yval = 6.5 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
11), yval = 4 });
 data1.Add(new DataSource() { xDate = new DateTime(2018, 1,
12), yval = 5.5 });
 return data1;
 }
 }
 }
}

```

```

 }
}
}
}

```

<!-- markdownlint-disable MD036 -->

## Category

You can assign category values to the sparkline by setting [valueType](#) to **Category**.

## C#HTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("1
30px").DataSource(ViewBag.category).Type(SparklineType.Column).XName("xval")
.YName("yval").ValueType(SparklineValueType.Category).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

## CATEGORY.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.category = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource

```

```

 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { xval = "Robert", yval = 60 });
 data2.Add(new DataSource() { xval = "Andrew", yval = 65 });
 data2.Add(new DataSource() { xval = "Suyama", yval = 70 });
 data2.Add(new DataSource() { xval = "Michael", yval = 80 });
 data2.Add(new DataSource() { xval = "Janet", yval = 55 });
 data2.Add(new DataSource() { xval = "Davolio", yval = 90 });
 data2.Add(new DataSource() { xval = "Fuller", yval = 75 });
 data2.Add(new DataSource() { xval = "Nancy", yval = 85 });
 return data2;
 }
 }
}

```

## Numeric

You can assign numeric values to the sparkline by setting [valueType](#) to **Numeric**.

## CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("130px").DataSource(ViewBag.numeric).Type(SparklineType.Column).XName("x").YName("yval").ValueType(SparklineValueType.Numeric).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

## NUMERIC.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;

```

```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.numeric = DataSource.GetNumericData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetNumericData()
 {
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { x = 1, yval = 190});
 data3.Add(new DataSource() { x = 2, yval = 165});
 data3.Add(new DataSource() { x = 3, yval = 158});
 data3.Add(new DataSource() { x = 4, yval = 175});
 data3.Add(new DataSource() { x = 5, yval = 200});
 data3.Add(new DataSource() { x = 6, yval = 180});
 data3.Add(new DataSource() { x = 7, yval = 210});
 return data3;
 }
 }
 }
}

```

<!-- markdownlint-disable MD036 -->

### Change min and max values of axis

You can change the min and max values of x-axis by setting the [minX](#) and [maxX](#) values to the [axisSettings](#) property. You can also change the min and max values of y-axis by setting the [minY](#) and [maxY](#) values to the [axisSettings](#) property.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("130px").AxisSettings(axis =>
 axis.Value(25).DataSource(ViewBag.numeric).Type(SparklineType.Column).XName("x").YName("yval").ValueType(SparklineValueType.Numeric).Render()
)
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 }
</style>

```

```

 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### MINMAX.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.numeric = DataSource.GetNumericData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetNumericData()
 {
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { x = 0, yval = 50});
 data3.Add(new DataSource() { x = 1, yval = 30});
 data3.Add(new DataSource() { x = 2, yval = 20});
 data3.Add(new DataSource() { x = 3, yval = 30});
 data3.Add(new DataSource() { x = 4, yval = 50});
 data3.Add(new DataSource() { x = 5, yval = 40});
 data3.Add(new DataSource() { x = 6, yval = 20});
 data3.Add(new DataSource() { x = 7, yval = 10 });
 data3.Add(new DataSource() { x = 8, yval = 30 });
 data3.Add(new DataSource() { x = 9, yval = 10 });
 data3.Add(new DataSource() { x = 10, yval = 40 });
 return data3;
 }
 }
 }
}

```

```
}

```

### Change value of axis

You can set horizontal axis line value of the sparkline by setting [value](#) to the [axisSettings](#) property. The following code example shows this.

#### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("130px").AxisSettings(axis =>
axis.MinY(0).MaxY(150)).DataSource(ViewBag.numeric).Type(SparklineType.Column).XName("x").YName("yval").ValueType(SparklineValueType.Numeric).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

#### VALUE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.numeric = DataSource.GetNumericData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 }
 }
}
```



```

public int x;
public string yval;
public static List<DataSource> GetNumericData()
{
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { x = 0, yval = 50});
 data3.Add(new DataSource() { x = 1, yval = 30});
 data3.Add(new DataSource() { x = 2, yval = 20});
 data3.Add(new DataSource() { x = 3, yval = 30});
 data3.Add(new DataSource() { x = 4, yval = 50});
 data3.Add(new DataSource() { x = 5, yval = 40});
 data3.Add(new DataSource() { x = 6, yval = 20});
 data3.Add(new DataSource() { x = 7, yval = 10 });
 data3.Add(new DataSource() { x = 8, yval = 30 });
 data3.Add(new DataSource() { x = 9, yval = 10 });
 data3.Add(new DataSource() { x = 10, yval = 40 });
 return data3;
}
}
}

```

### Axis line customization

Axis of the sparkline can be collapsed using the [visible](#) property in [lineSettings](#); this is not applicable for win-loss. You can customize the [color](#), [width](#), [opacity](#), and [dashArray](#) of axis line.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("130px").AxisSettings(axis => axis.LineSettings(new SparklineLineSettings {
Visible = true, Color = "#ff14ae", DashArray = "5"
})).DataSource(ViewBag.numeric).Type(SparklineType.Column).XName("x").YName("yval").ValueType(SparklineValueType.Numeric).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### AXIS\_LINE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.numeric = DataSource.GetNumericData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetNumericData()
 {
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { x = 0, yval = 50});
 data3.Add(new DataSource() { x = 1, yval = 30});
 data3.Add(new DataSource() { x = 2, yval = 20});
 data3.Add(new DataSource() { x = 3, yval = 30});
 data3.Add(new DataSource() { x = 4, yval = 50});
 data3.Add(new DataSource() { x = 5, yval = 40});
 data3.Add(new DataSource() { x = 6, yval = 20});
 data3.Add(new DataSource() { x = 7, yval = 10 });
 data3.Add(new DataSource() { x = 8, yval = 30 });
 data3.Add(new DataSource() { x = 9, yval = 10 });
 data3.Add(new DataSource() { x = 10, yval = 40 });
 return data3;
 }
 }
 }
}

```

### Special points customization

You can customize the points by initializing the point colors. The customization options allows to differentiate the [start](#), [end](#), [positive](#), [negative](#), and [low](#) points. This customization is only applicable for line, column, and area type sparklines.

<!-- markdownlint-disable MD036 -->

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("1
30px").XName("xval").YName("yval").Type(SparklineType.Column).ValueType(Spar

```

```

klineValueType.Category).HighPointColor("blue").LowPointColor("orange").StartPointColor("green").EndPointColor("green").NegativePointColor("red").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### CUSTOM.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = DataSource.GetCategoryData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetCategoryData()
 {
 List<DataSource> data2 = new List<DataSource>();
 data2.Add(new DataSource() { x = 0, xval = "AUDI", yval = 1 });
 data2.Add(new DataSource() { x = 1, xval = "BMW", yval = 5 });
 data2.Add(new DataSource() { x = 2, xval = "BUICK", yval = -1 });
 }
 }
 }
}

```

```

-6 });
 data2.Add(new DataSource() { x = 3, xval = "CETROEN", yval =
= 0 });
 data2.Add(new DataSource() { x = 4, xval = "CHEVROLET", yval
});
 data2.Add(new DataSource() { x = 5, xval = "FIAT", yval = 1
});
 data2.Add(new DataSource() { x = 6, xval = "FORD", yval = -2
});
 data2.Add(new DataSource() { x = 7, xval = "HONDA", yval = 7
});
 data2.Add(new DataSource() { x = 8, xval = "HYUNDAI", yval =
-9 });
 data2.Add(new DataSource() { x = 9, xval = "JEEP", yval = 0
});
 data2.Add(new DataSource() { x = 10, xval = "KIA", yval = -
10 });
 data2.Add(new DataSource() { x = 11, xval = "MAZDA", yval =
3 });
 return data2;
}
}
}
}

```

### Tie point color

Tie point color is used to configure the win-loss series type sparkline's y-value point color. The following code sample shows the tie point color of sparkline series.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("150px").Width("1
30px").Type(SparklineType.WinLoss).ValueType(SparklineValueType.Numeric).Tie
PointColor("blue").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### TIE POINT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 12, 15, -10, 13, 15, 6, -12, 17,
13, 0, 8, -10 };
 return View();
 }
 }
}

```

## Range Band

This section explains how to customize the sparkline with multiple range bands.

### Range band customization

The range band feature is used to highlight a particular range along with the y-axis using the [startRange](#) and [endRange](#) properties. You can also customize the [color](#) and [opacity](#) of the range band.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("1
50px").LineWidth(2).Fill("#0d3c9b").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
function loaded(args)
{
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.rangeBandSettings[0] = {
 startRange: 1,
 endRange: 3,
 color: "#bfd4fc",
 opacity: 0.4
 };
 args.sparkline.refresh();
}

```

```
</script>
```

### RANGE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 0, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}
```

### Multiple range band customization

You can define multiple range bands to a sparkline as shown in the following code sample.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("150px").LineWidth(2).Fill("#0d3c9b").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
function loaded(args)
{
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.rangeBandSettings[0] = {
 startRange: 1,
 endRange: 3,
 color: "#bfd4fc",
 opacity: 0.4
 };
 args.sparkline.rangeBandSettings[1] = {
 startRange: 4,
 endRange: 5,
```

```

 color: 'red',
 opacity: 0.4
 };
 args.sparkline.refresh();
}
</script>

```

## MULTI RANGE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 0, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

## Markers

This section explains how to add markers to the sparklines.

### Adding marker to the sparkline

To add marker to the sparkline, specify the [visible](#) of [markerSettings](#) as following values. The [visible](#) will accept multiple values too.

- All - Enables markers for all points.
- Start - Enables marker for the start point.
- End - Enables marker for the end point.
- High - Enables marker for the high point.
- Low - Enables marker for the low point.
- Negative - Enables markers for the negative points.

The following code example shows enabling markers for all points.

## CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(7)).DataSource(ViewBag.sparkData).Render()
</div>

```

```

<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.markerSettings = {
 visible: ['All']
 };
 args.sparkline.refresh();
 }
</script>

```

### ALLPOINT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 0, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

### Adding marker to special point

In sparkline, markers can be enabled for particular points such as the start, end, low, high, or negative. The following code examples shows enabling markers for the high and low points.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(7)).HighPointColor("blue").LowPointColor("red").DataSource(ViewBag.sparkData).Render()
</div>

```



```

<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.markerSettings = {
 visible: ['high', 'Low']
 };
 args.sparkline.refresh();
 }
</script>

```

### SPECIALPOINT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 3, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

### Customizing markers

Sparkline markers can be customized in terms of fill color, border color, width, opacity, and size. The following code example shows customizing marker's fill, border, and size.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(7)).Fill("blue").DataSource(ViewBag.sparkData).Render()
</div>
<style>

```

```

 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
 </style>
 <script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.markerSettings = {
 visible: ['all'],
 size: 5,
 fill: 'white',
 border: { color: 'blue', width: 2 }
 };
 args.sparkline.refresh();
 }
 </script>

```

### CUSTOM.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 3, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

### Data Labels

Data labels are used to display values of data points to improve the readability.

#### Enable data label

To enable data label for sparkline series, provide [visible](#) of the [dataLabelSettings](#) as following possible values:

- All - Enables data label of all points.
- Start - Enables data label of the start point.
- End - Enables data label of the end point.

- High - Enables data label of the high point.
- Low - Enables data label of the low point.
- Negative - Enables data labels of the negative points.

The following example shows enabling sparkline data label for all points.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(7)).Fill("blue").DataLabelSettings(data => data.Visible(ViewBag.sparkVisible)).DataSource(ViewBag.sparkData).Render()
</div>
<style>
.spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
}
</style>
<script>
function loaded(args)
{
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
}
</script>
```

### ENABLE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkVisible = new string[] { "All" };
 ViewBag.sparkData = new int[] { 0, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}
```

### Customize data label

Data labels can be customized using the fill, border, opacity, and text Style. The following code example shows customizing data label border, text color, and fill color.

#### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(7)).Fill("blue").DataLabelSettings(data => data.Fill("blue").Opacity(0.4).Visible(ViewBag.sparkVisible)).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

#### CUSTOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkVisible = new string[] { "All" };
 ViewBag.sparkData = new int[] { 0, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}
```

### Format data label text

The text of data labels can be formatted using the [format](#) API in the sparkline [dataLabelSettings](#). By default, data label shows the y-value of point. The following code example shows how to display x and y-values for points.

#### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("500px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(7)).DataLabelSettings(data => data.Format("{xval} : {yval}").Visible(ViewBag.sparkVisible).Fill("blue").Opacity(0.4).Fill("blue").DataSource(ViewBag.numeric).XName("xval").YName("yval").ValueType(SparklineValueType.Category)).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

#### FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkVisible = new string[] { "All" };
 ViewBag.numeric = DataSource.GetNumericData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
```

```

public double yval;
public int x;
public string xval;
public static List<DataSource> GetNumericData()
{
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { xval = "Mon", yval = 3});
 data3.Add(new DataSource() { xval = "Tue", yval = 5});
 data3.Add(new DataSource() { xval = "Wed", yval = 2});
 data3.Add(new DataSource() { xval = "Thu", yval = 4});
 data3.Add(new DataSource() { xval = "Fri", yval = 6});
 return data3;
}
}
}
}

```

## User interactions

Sparkline has two user interaction features: tooltip and tracker line.

### Tooltip

The sparkline provides options to display details about values of data points through tooltips when hovering the mouse over data point. To use tooltip in sparkline, inject the [SparklineTooltip](#) module to sparkline using the inject method.

The following code example shows enabling tooltip for sparkline with format.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("500px").AxisSettings(axis => axis.MinX(-1).MaxX(8)).Fill("blue").ValueType(SparklineValueType.Category).XName("xval").YName("yval").TooltipSettings(tool => tool.Visible(true).Format("${xval} : ${yval}")).DataSource(ViewBag.data).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### TOOLTIP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.data = DataSource.GetData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetData()
 {
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { xval = "Mon", yval = 3});
 data3.Add(new DataSource() { xval = "Tue", yval = 5});
 data3.Add(new DataSource() { xval = "Wed", yval = 2});
 data3.Add(new DataSource() { xval = "Thu", yval = 4});
 data3.Add(new DataSource() { xval = "Fri", yval = 6});
 return data3;
 }
 }
 }
}

```

### Tooltip customization

The fill color, text styles, format, and border of the tooltip can be customized. The following code example shows customization of tooltip's fill color and text style.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("500px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(8)).Fill("#033e96").ValueType(SparklineValueType.Category).XName("xval").YName("yval").TooltipSettings(tool =>
tool.Visible(true).Format("${xval} : ${yval}").Fill("#033e96")).DataSource(ViewBag.data).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 }

```

```
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

### TOOLTIP-CUSTOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.data = DataSource.GetData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetData()
 {
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { xval = "Mon", yval = 3});
 data3.Add(new DataSource() { xval = "Tue", yval = 5});
 data3.Add(new DataSource() { xval = "Wed", yval = 2});
 data3.Add(new DataSource() { xval = "Thu", yval = 4});
 data3.Add(new DataSource() { xval = "Fri", yval = 6});
 return data3;
 }
 }
 }
}
```



*Tooltip template*

Sparkline tooltip has template support. By using tooltip template, you can customize tooltips. The following code example shows more customization options provided to `sparktooltip` class that is used in tooltip template div. Using this template, images also can be added to tooltip.

**CSHTML**

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("500px").AxisSettings(axis => axis.MinX(-1).MaxX(7).MinY(-1).MaxY(8)).Fill("#033e96").ValueType(SparklineValueType.Category).XName("xval").YName("yval").TooltipSettings(new SparklineSparklineTooltipSettings { Visible = true, Template = "#tooltip" }).DataSource(ViewBag.data).Render()
</div>
<div id="tooltip">
 ${xval} : ${yval}
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
 #tooltip {
 border-radius: 5px;
 background: #008cff;
 color: #FFFFFF !important;
 font-size: 16px;
 font-style: italic;
 padding: 8px;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

**TOOLTIP\_TEMPLATE.CS**

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
```

```

{
 public IActionResult Index()
 {
 ViewBag.data = DataSource.GetData();
 return View();
 }
 public class DataSource
 {
 public DateTime xDate;
 public double yval;
 public int x;
 public string xval;
 public static List<DataSource> GetData()
 {
 List<DataSource> data3 = new List<DataSource>();
 data3.Add(new DataSource() { xval = "Mon", yval = 3});
 data3.Add(new DataSource() { xval = "Tue", yval = 5});
 data3.Add(new DataSource() { xval = "Wed", yval = 2});
 data3.Add(new DataSource() { xval = "Thu", yval = 4});
 data3.Add(new DataSource() { xval = "Fri", yval = 6});
 return data3;
 }
 }
}

```

### Track line

The track line tracks data points that are closer to the mouse position or touch contact.

To enable track lines for sparkline, specify the [visible](#) option of [trackLineSettings](#) to true. Based on theme, tracker color will be changed. The default value of tracker color is black.

To use track line in sparkline, inject the [SparklineTooltip](#) module to sparkline using the inject method.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("500px").AxisSettings(axis => axis.MinX(-1).MaxX(46).MinY(-1).MaxY(10)).Fill("#033e96").TooltipSettings(tool => tool.Visible(true).TrackLineSettings(new SparklineTrackLineSettings { Visible = true, Color = "#033e96", Width = 1})).DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {

```

```

 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### TOOLTIP\_TRACKLINE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 5, 3, 4, 6, 8, 7, 9, 1, 3, 5, 3,
4, 6, 8, 7, 9, 1, 3, 5, 2, 4, 6, 7, 9, 5, 8, 3, 6, 1, 7, 4, 2, 5, 2, 4, 6,
7, 9, 5, 8, 3, 6, 1, 7, 4, 2 };
 return View();
 }
 }
}

```

### Appearance

The appearance of the sparkline can be customized using margin, container Area border, and container Area background.

#### Sparkline border

The [containerArea border](#) of the sparkline is used to render border to cover sparkline area.

The following code example shows the sparkline with overall border.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
@Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").ContainerArea(con => con.Border(new SparklineSparklineBorder { Color = "#033e96", Width = 2 })).Border(bod => bod.Color("#033e96").Width(1)).Type(SparklineType.Area).Fill("#b2cfff").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }

```

```

</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

## BORDER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 3, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

## Sparkline padding

Padding is used to specify padding value between container and sparkline. By default, padding value of the sparkline is 5. Sparkline [padding](#) values are specified by the left, right, top, and bottom.

The following code example shows the sparkline with overall padding is set to 20.

## CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").ContainerArea(con => con.Border(new SparklineSparklineBorder { Color = "#033e96", Width = 2 })).Padding(pad => pad.Left(20).Right(20).Bottom(20).Top(20)).Border(bod => bod.Color("#033e96").Width(1)).Type(SparklineType.Area).Fill("#b2cfff").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }

```

```

</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

## PADDING.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 3, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

## Sparkline area customization

The background color of the sparkline area can be customized using the [containerArea background](#) color. By default, the sparkline background color is transparent.

## CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").ContainerArea(con => con.Background("#eff1f4").Border(new SparklineSparklineBorder { Color = "#033e96", Width = 2 })).Padding(pad => pad.Left(20).Right(20).Bottom(20).Top(20)).Border(bod => bod.Color("#033e96").Width(1)).Type(SparklineType.Area).Fill("#b2cfff").DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>

```

```
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>
```

### AREA\_CUSTOM.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 3, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}
```

### Sparkline theme

Datalabel and track line colors of the sparkline will be changed based on theme. For example, for dark theme, the color of datalabel and track line should be white; for light theme, their value should be black. The possible values for sparkline theme are [Material](#), [Fabric](#), [Bootstrap](#), and [Highcontrast](#).

The following code example shows the color for datalabel and track line is set to white for dark theme.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").Theme(SparklineTheme.Highcontrast).DataLabelSettings(data =>
 data.Visible(ViewBag.sparkVisible)).TooltipSettings(tool =>
 tool.TrackLineSettings(new SparklineTrackLineSettings { Visible = true
 })).AxisSettings(axis => axis.MinX(-1).MaxX(7)).LineWidth(3).Border(bor =>
 bor.Color("transparent").Width(2)).Type(SparklineType.Line).Fill("#007dd1").
 DataSource(ViewBag.sparkData).Render()
</div>
<style>
 .spark {
 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
```

```

 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### THEME.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 3, 6, 4, 1, 3, 2, 5 };
 return View();
 }
 }
}

```

### Localization

The sparkline control supports localization. The default culture for localization is **en-US**. You can change the culture using the **setCulture** method.

#### Tooltip format

Sparkline tooltip supports localization. The following code sample shows tooltip text with currency format based on culture.

### CSHTML

```

@using Syncfusion.EJ2;
<div class="spark" align="center">
 @Html.EJS().Sparkline("sparkline").Loaded("loaded").Height("200px").Width("350px").TooltipSettings(tool =>
 tool.Visible(true)).Format("c0").UseGroupingSeparator(true).LineWidth(3).Padding(pad => pad.Left(20).Right(20).Bottom(20).Top(20)).Border(bod =>
 bod.Color("#033e96").Width(2)).Type(SparklineType.Area).Fill("#b2cfff").DataSource(ViewBag.sparkData).Render()
 </div>
<style>
 .spark {

```

```

 border: 1px solid rgb(209, 209, 209);
 border-radius: 2px;
 width: 100%;
 height: 100%;
 }
</style>
<script>
 function loaded(args)
 {
 window.sparkline = args.sparkline;
 args.sparkline.loaded = null;
 args.sparkline.refresh();
 }
</script>

```

### TOOLTIP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 ViewBag.sparkData = new int[] { 30000, 60000, 40000, 10000,
30000, 20000, 50000 };
 return View();
 }
 }
}

```

### Accessibility in ASP.NET MVC Sparkline component

The Sparkline component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Sparkline component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |



```

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

The Sparkline component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Sparkline component:

- `img (role)`
- `aria-label (attribute)`
- `aria-hidden (attribute)`

### Keyboard interaction

The Sparkline component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Sparkline component.

| **Press** | **To do this** |

| --- | --- |

| **Ctrl + P** | Prints the Sparkline. |

### Ensuring accessibility

The Sparkline component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Sparkline component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Sparkline component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Migration from Essential JS 1

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

#### Sparkline Types

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Type| **Property:** *type*<br/><br/>

@(Html.EJ().Sparkline("container").Type(SparklineType.Column))| **Property:** *type*<br/><br/>

@Html.EJS().Sparkline("container").Type(Syncfusion.EJ2.Charts.SparklineType.Area).Render() |

#### Databinding

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Datasource| **Property:** *dataSource*<br/><br/>

@(Html.EJ().Sparkline("container").DataSource(ViewBag.SparklineData)) | **Property:**

*dataSource*<br/><br/>

@Html.EJS().Sparkline("container").DataSource(ViewBag.datasource).Render() |

|Binding X values with datasource| **Property:** *xName*<br/><br/>

@(Html.EJ().Sparkline("container").XName("Month")) | **Property:** *xName*<br/><br/>

@Html.EJS().Sparkline("container").XName("Month").Render() |

|Binding Y values with datasource| **Property:** *yName*<br/><br/>

@(Html.EJ().Sparkline("container").YName("Sales")) | **Property:** *yName*<br/><br/>

@Html.EJS().Sparkline("container").YName("Sales").Render() |

#### Markers

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Enable markers| **Property:** *markerSettings.visible*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Visible(true))) | **Property:**

*markerSettings.visible*<br/><br/> @Html.EJS().Sparkline("container").MarkerSettings(ms =>ms.Visible(new string[] { "All" })).Render() |

|Color| **Property:** *markerSettings.fill*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Fill("red"))) | **Property:** *markerSettings.fill*<br/><br/> @Html.EJS().Sparkline("container").MarkerSettings(ms =>ms.Fill("red")).Render()|

|Size| **Property:** *markerSettings.width*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Width(10))) | **Property:** *markerSettings.size*<br/><br/> @Html.EJS().Sparkline("container").MarkerSettings(ms =>ms.Size(10)).Render()|

|Opacity| **Property:** *markerSettings.opacity*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Opacity(0.5))) | **Property:** *markerSettings.opacity*<br/><br/> @Html.EJS().Sparkline("container").MarkerSettings(ms =>ms.Opacity(0.5)).Render()|

|Border color| **Property:** *markerSettings.border.color*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Border(br=> br.Color("black")))| **Property:** *markerSettings.border.color*<br/><br/> @Html.EJS().Sparkline("container").MarkerSettings(ms =>ms.Border(br=> br.Color("Black"))).Render()|

|Border width| **Property:** *markerSettings.border.width*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Border(br=> br.Width(1))) | **Property:** *markerSettings.border.width*<br/><br/> @Html.EJS().Sparkline("container").MarkerSettings(ms =>ms.Border(br=> br.Width(1))).Render()|

|Border opacity| **Property:** *markerSettings.border.opacity*<br/><br/>

@(Html.EJ().Sparkline("container").MarkerSettings(mr => mr.Border(br=> br.Opacity(0.5)))) | Not applicable|

## Data labels

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Enable data labels| Not applicable | **Property:** *dataLabelSettings.visible*<br/><br/>

@Html.EJS().Sparkline("container").DataLabelSettings(dl => dl.Visible(new string[] { "All" })).Render()|

|Color| Not applicable | **Property:** *dataLabelSettings.fill*<br/><br/>

@Html.EJS().Sparkline("container").DataLabelSettings(dl => dl.Fill("red")).Render()|

|Opacity| Not applicable | **Property:**

*dataLabelSettings.opacity*<br/><br/>@Html.EJS().Sparkline("container").DataLabelSettings(dl => dl.Opacity(0.5)).Render() |

|Border color| Not applicable | **Property:** *dataLabelSettings.border.color*<br/><br/>

@Html.EJS().Sparkline("container").DataLabelSettings(dl => dl.Border(ViewBag.border)).Render()<br/><br/>ViewBag.border = new {color="red"}|

| Border width | Not applicable | **Property:** *dataLabelSettings.border.width*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.Border(ViewBag.border)).Render() | *ViewBag.border = new {width=2}* |

| Format | Not applicable | **Property:** *dataLabelSettings.format*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl => dl.Format("{xval}:"  
 "{yval}")).Render() |

| Horizontal Offset | Not applicable | **Property:** *dataLabelSettings.offset.x*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.Offset(ViewBag.offset)).Render() | *ViewBag.offset = new {x=100}* |

| Vertical Offset | Not applicable | **Property:** *dataLabelSettings.offset.y*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.Offset(ViewBag.offset)).Render() | *ViewBag.offset = new {y=100}* |

| Font color | Not applicable | **Property:** *dataLabelSettings.textStyle.color*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.TextStyle(ViewBag.textStyle)).Render() | *ViewBag.textStyle = new {color="green"}* |

| Font family | Not applicable | **Property:** *dataLabelSettings.textStyle.fontFamily*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.TextStyle(ViewBag.textStyle)).Render() | *ViewBag.textStyle = new  
 {fontFamily="Arial"}* |

| Font style | Not applicable | **Property:** *dataLabelSettings.textStyle.fontStyle*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.TextStyle(ViewBag.textStyle)).Render() | *ViewBag.textStyle = new  
 {fontStyle="normal"}* |

| Font weight | Not applicable | **Property:** *dataLabelSettings.textStyle.fontWeight*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.TextStyle(ViewBag.textStyle)).Render() | *ViewBag.textStyle = new  
 {fontWeight="bold"}* |

| Font opacity | Not applicable | **Property:** *dataLabelSettings.textStyle.opacity*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.TextStyle(ViewBag.textStyle)).Render() | *ViewBag.textStyle = new {opacity=0.5}* |

| Font size | Not applicable | **Property:** *dataLabelSettings.textStyle.fontSize*  
 @Html.EJS().Sparkline("container").DataLabelSettings(dl =>  
 dl.TextStyle(ViewBag.textStyle)).Render() | *ViewBag.textStyle = new {fontSize="12px"}* |

## Range band

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Color | **Property:**  
*rangeBandSettings.color* | @Html.EJ().Sparkline("container").RangeBandSettings(range  
 => range.Color("red")) | **Property:** *rangeBandSettings.color*  
 @Html.EJS().Sparkline("container").RangeBandSettings(rbs => rbs.Color("red")).Render() |

|Opacity| **Property:** *rangeBandSettings.opacity*<br/><br/>  
 <@Html.EJ().Sparkline("container").RangeBandSettings(range => range.Opacity(0.4)) | **Property:** *rangeBandSettings.opacity*<br/><br/> @Html.EJS().Sparkline("container").RangeBandSettings(rbs => rbs.Opacity(0.5)).Render()|

|Start range| **Property:** *rangeBandSettings.startRange*<br/><br/>  
 @Html.EJ().Sparkline("container").RangeBandSettings(range => range.StartRange(4)) | **Property:** *rangeBandSettings.startRange*<br/><br/>  
 @Html.EJS().Sparkline("container").RangeBandSettings(rbs => rbs.StartRange(5)).Render()|

|End range| **Property:** *rangeBandSettings.endRange*<br/><br/>  
 @Html.EJ().Sparkline("container").RangeBandSettings(range => range.EndRange(30)) | **Property:** *rangeBandSettings.endRange*<br/><br/>  
 @Html.EJS().Sparkline("container").RangeBandSettings(rbs => rbs.EndRange(15)).Render()|

### Special points customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|High point color| **Property:** *highPointColor*<br/><br/>  
 @Html.EJ().Sparkline("container").HighPointColor("Blue")) | **Property:** *highPointColor*<br/><br/>  
 @Html.EJS().Sparkline("container").HighPointColor("red").Render()|

|Low point color| **Property:** *lowPointColor*<br/><br/>  
 @Html.EJ().Sparkline("container").LowPointColor("Orange")) | **Property:** *lowPointColor*<br/><br/>  
 @Html.EJS().Sparkline("container").LowPointColor("blue").Render()|

|Negative point color| **Property:** *negativePointColor*<br/><br/>  
 @Html.EJ().Sparkline("container").NegativePointColor("Red")) | **Property:** *negativePointColor*<br/><br/>  
 @Html.EJS().Sparkline("container").NegativePointColor("green").Render()|

|Start point color| **Property:** *startPointColor*<br/><br/>  
 @Html.EJ().Sparkline("container").StartPointColor("Green")) | **Property:** *startPointColor*<br/><br/>  
 <@Html.EJS().Sparkline("container").StartPointColor("black").Render()|

|End point color| **Property:** *endPointColor*<br/><br/>  
 @Html.EJ().Sparkline("container").EndPointColor("Green")) | **Property:** *endPointColor*<br/><br/>  
 @Html.EJS().Sparkline("container").EndPointColor("orange").Render()|

|Tie point color| **Property:** *tiePointColor*<br/><br/>Not Applicable | **Property:** *tiePointColor*<br/><br/>  
 @Html.EJS().Sparkline("container").TiePointColor("grey").Render()|

### Axis customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Show axis line| **Property:** *axisSettings.visible*<br/><br/>  
 @Html.EJ().Sparkline("container").AxisLineSettings(as => as.Visible(true))) | **Property:**

*axisSettings.lineSettings.visible*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.LineSettings(ls => ls.Visible(true))).Render()

| Line color | **Property:** *axisSettings.color*  
@Html.EJ().Sparkline("container").AxisLineSettings(as => as.Color("#ff14ae")) | **Property:** *axisSettings.lineSettings.color*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.LineSettings(ls => ls.Color("red"))).Render()

| Line width | **Property:** *axisSettings.width*  
@Html.EJ().Sparkline("container").AxisLineSettings(as => as.Width(2)) | **Property:** *axisSettings.lineSettings.width*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.LineSettings(ls => ls.Width(2))).Render()

| Dash array | **Property:** *axisSettings.dashArray*  
@Html.EJ().Sparkline("container").AxisLineSettings(as => as.DashArray("5,3")) | **Property:** *axisSettings.lineSettings.dashArray*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.LineSettings(ls => ls.DashArray("5,3"))).Render()

| X axis minimum value | Not applicable | **Property:** *axisSettings.minX*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.MinX(0)).Render()

| X axis maximum value | Not applicable | **Property:** *axisSettings.maxX*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.MaxX(100)).Render()

| Y axis minimum value | Not applicable | **Property:** *axisSettings.minY*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.MinY(0)).Render()

| Y axis maximum value | Not applicable | **Property:** *axisSettings.maxY*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.MaxY(100)).Render()

| Horizontal axis line position | Not applicable | **Property:** *axisSettings.value*  
@Html.EJS().Sparkline("container").AxisSettings(axis => axis.Value(10)).Render()

#### Appearance customization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Background color | **Property:** *background*  
@Html.EJ().Sparkline("container").Background("gray") | **Property:** *containerArea.background*  
@Html.EJS().Sparkline("container").ContainerArea(ca => ca.Background("red")).Render()

| Border color | Not applicable | **Property:** *containerArea.border.color*  
@Html.EJS().Sparkline("container").ContainerArea(ca => ca.Border(ViewBag.border)).Render()  
ViewBag.border = new {color="green"}

| Border width | Not applicable | **Property:** *containerArea.border.width*  
@Html.EJS().Sparkline("container").ContainerArea(ca => ca.Border(ViewBag.border)).Render()  
ViewBag.border = new {color="green"}



|Series color| **Property:** `fill`  
**Property:** `@Html.EJ().Sparkline("container").Fill("gray")` | **Property:** `fill`  
**Property:** `@Html.EJS().Sparkline("container").Fill("green").Render()` |

|Series opacity| **Property:** `opacity`  
**Property:** `@Html.EJ().Sparkline("container").Opacity(0.5)` | **Property:** `opacity`  
**Property:** `@Html.EJS().Sparkline("container").Opacity(0.5).Render()` |

|Line series width| **Property:** `width`  
**Property:** `@Html.EJ().Sparkline("container").Width(2)` | **Property:** `lineWidth`  
**Property:** `@Html.EJS().Sparkline("container").LineWidth(2).Render()` |

|Series border color| **Property:** `border.color`  
**Property:** `@Html.EJ().Sparkline("container").Border(border=>border.Color("green"))` | **Property:** `border.color`  
**Property:** `@Html.EJS().Sparkline("container").Border(br=>br.Color("red")).Render()` |

|Series border width| **Property:** `border.width`  
**Property:** `@Html.EJ().Sparkline("container").Border(border=>border.Width(2))` | **Property:** `border.width`  
**Property:** `@Html.EJS().Sparkline("container").Border(br=>br.Width(2)).Render()` |

|Series palette| **Property:** `palette`  
**Property:** `@Html.EJ().Sparkline("container").Palette(ViewBag.palettes)`  
**Property:** `@Html.EJS().Sparkline("container").Palette(ViewBag.palettes).Render()`  
**Property:** `ViewBag.palettes = new string[] { "red", "green", "orange", "blue" }` | **Property:** `palette`  
**Property:** `@Html.EJS().Sparkline("container").Palette(ViewBag.palettes).Render()`  
**Property:** `ViewBag.palettes = new string[] { "red", "green", "orange", "blue" }` |

|Theme| **Property:** `theme`  
**Property:** `@Html.EJ().Sparkline("container").Theme(SparkTheme.FlatDark)` | **Property:** `theme`  
**Property:** `@Html.EJS().Sparkline("container").Theme("Material").Render()` |

|Width| **Property:** `size.width`  
**Property:** `@Html.EJ().Sparkline("container").Width("300px")` | **Property:** `width`  
**Property:** `@Html.EJS().Sparkline("container").Width("300px").Render()` |

|Height| **Property:** `size.height`  
**Property:** `@Html.EJ().Sparkline("container").Height("300px")` | **Property:** `height`  
**Property:** `@Html.EJS().Sparkline("container").Height("300px").Render()` |

### Tooltip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

|Show tooltip| **Property:** `tooltip.visible`  
**Property:** `@Html.EJ().Sparkline("container").Tooltip(tooltip => tooltip.Visible(true))` | **Property:** `tooltipSettings.visible`  
**Property:** `@Html.EJS().Sparkline("container").TooltipSettings(tooltip => tooltip.Visible(true)).Render()` |

|Background| **Property:** `tooltip.fill`  
**Property:** `@Html.EJ().Sparkline("container").Tooltip(tooltip => tooltip.Fill("red"))` | **Property:** `tooltipSettings.fill`  
**Property:** `@Html.EJS().Sparkline("container").TooltipSettings(tooltip => tooltip.Fill("red")).Render()` |

|Format| Not applicable | **Property:** `tooltipSettings.format`  
**Property:** `@Html.EJS().Sparkline("container").TooltipSettings(tooltip => tooltip.Format("{xval}: {yval}"))` | **Property:** `tooltipSettings.format`  
**Property:** `@Html.EJS().Sparkline("container").TooltipSettings(tooltip => tooltip.Format("{xval}: {yval}"))` |

|Template| **Property:** `tooltip.template`  
**Property:** `@Html.EJ().Sparkline("container").Tooltip(tooltip =>`

```

tooltip.Template("tooltip"))

<div id="tooltip"></div>
<div>#point.x#</div></br> <div>#point.y#</div></br></div>| Property:
tooltipSettings.template

 @Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.Template("tooltip")).Render()

<div id="tooltip">${x} : ${y}</div>|

|Font color| Property: tooltip.font.color

@(Html.EJ().Sparkline("container").Tooltip(tooltip => tooltip.Font(font=>font.Color("red"))))
| Property: tooltipSettings.textStyle.color

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TextStyle(ViewBag.font)).Render()

ViewBag.font = new { color="gray"};|

|Font opacity| Property: tooltip.font.opacity

@(Html.EJ().Sparkline("container").Tooltip(tooltip => tooltip.Font(font=>font.Opacity(0.5))))
| Property: tooltipSettings.textStyle.opacity

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TextStyle(ViewBag.font)).Render()

ViewBag.font = new { opacity=0.5};|

|Font size| Property: tooltip.font.size

 @(Html.EJ().Sparkline("container").Tooltip(tooltip
=> tooltip.Font(font=>font.Size('12px')))) | Property: tooltipSettings.textStyle.size

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TextStyle(ViewBag.font)).Render()

ViewBag.font = new { size="14px"};|

|Font family| Property: tooltip.font.fontFamily

@(Html.EJ().Sparkline("container").Tooltip(tooltip =>
tooltip.Font(font=>font.FontFamily('Algerian')))) | Property:
tooltipSettings.textStyle.fontFamily

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TextStyle(ViewBag.font)).Render()

ViewBag.font = new { fontFamily="Arial"};|

|Font style| Property: tooltip.font.fontStyle

@(Html.EJ().Sparkline("container").Tooltip(tooltip =>
tooltip.Font(font=>font.FontStyle('Italic')))) | Property: tooltipSettings.textStyle.fontStyle

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TextStyle(ViewBag.font)).Render()

ViewBag.font = new { fontStyle="normal"};|

|Font weight| Property: tooltip.font.fontWeight

@(Html.EJ().Sparkline("container").Tooltip(tooltip =>
tooltip.Font(font=>font.FontWeight('Lighter')))) | Property:
tooltipSettings.textStyle.fontWeight

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TextStyle(ViewBag.font)).Render()

ViewBag.font = new { fontWeight="bold"};|

|Enable track line| Not applicable | Property: tooltipSettings.trackLineSettings.visible

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TrackLineSettings(ts=>ts.Visible(true))).Render()|

|Track line color| Not applicable | Property: tooltipSettings.trackLineSettings.color

@Html.EJS().Sparkline("container").TooltipSettings(tooltip =>
tooltip.TrackLineSettings(ts=>ts.Color("red"))).Render()|

```



|Track line width| Not applicable | **Property:** *tooltipSettings.trackLineSettings.width*<br/><br/>  
 @Html.EJS().Sparkline("container").TooltipSettings(tooltip =>  
 tooltip.TrackLineSettings(ts=>ts.Width(2)).Render() |

### Rendering

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Enable canvas rendering| **Property:** *enableCanvasRendering*<br/><br/>  
 @Html.EJ().Sparkline("container").EnableCanvasRendering(true)) | Not applicable |

### Localization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Localization| **Property:** *locale*<br/><br/> @Html.EJ().Sparkline("container").Locale("en-US")) |  
**Property:** *type*<br/><br/> @Html.EJS().Sparkline("container").Locale("en-US").Render() |

### Methods

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Dynamically updating sparkline| **Method:** *redraw*<br/><br/> var sparkline =  
 \$("#container").ejSparkline("instance");<br/>sparkline.redraw(); | **Method:** *refresh*<br/><br/> var  
 sparkline = document.getElementById("container").ej2\_instances[0];<br/>sparkline.refresh(); |

### Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

|Load| **Event:** *load*<br/><br/> @Html.EJ().Sparkline("container").Load("load")<br/><br/>function  
 load(args) { } | **Event:** *load*<br/><br/> `@Html.EJS().Sparkline("container").Load("load").Render()  
 <br/><br/>function load(args) { }` |

|Load completed| **Event:** *loaded*<br/><br/>  
 @Html.EJ().Sparkline("container").Loaded("loaded")<br/><br/>function loaded(args) { } |  
**Event:** *loaded*<br/><br/>  
 @Html.EJS().Sparkline("container").Loaded("loaded").Render()<br/><br/>function loaded(args) {  
 } |

|Initialize tooltip| **Event:** *tooltipInitialize*<br/><br/>  
 @Html.EJ().Sparkline("container").TooltipInitialize("tooltipInitialize")<br/><br/>function  
 tooltipInitialize(args) { } | **Event:** *tooltipInitialize*<br/><br/>  
 @Html.EJS().Sparkline("container").TooltipInitialize("tooltipInitialize").Render()<br/><br/>functi  
 on tooltipInitialize(args) { } |

|Series rendering| **Event:** *seriesRendering*<br/><br/>  
 @Html.EJ().Sparkline("container").SeriesRendering("seriesRendering")<br/><br/>function  
 seriesRendering(args) { } | **Event:** *seriesRendering*<br/><br/>

```
@Html.EJS().Sparkline("container").SeriesRendering("seriesRendering").Render()
function seriesRendering(args) { }
```

```
|Region mouse move| Event: pointRegionMouseMove
@(Html.EJ().Sparkline("container").PointRegionMouseMove("pointRegionMouseMove"))
function pointRegionMove(args) { } | Event: pointRegionMouseMove
@Html.EJS().Sparkline("container").PointRegionMouseMove("pointRegionMouseMove").Render()
function pointRegionMouseMove(args) { }
```

```
|Region click| Event: pointRegionMouseClick
@(Html.EJ().Sparkline("container").PointRegionMouseClick("pointRegionMouseClick"))
function pointRegionClick(args) { } | Event: pointRegionMouseClick
@Html.EJS().Sparkline("container").PointRegionMouseClick("pointRegionMouseClick").Render()
function pointRegionMouseClick(args) { }
```

```
|Mouse move| Event: sparklineMouseMove
@(Html.EJ().Sparkline("container").SparklineMouseMove("sparklineMouseMove"))
function mouseMove(args) { } | Event: sparklineMouseMove
@Html.EJS().Sparkline("container").SparklineMouseMove("sparklineMouseMove").Render()
function sparklineMouseMove(args) { }
```

```
|Mouse leave| Event: sparklineMouseLeave
@(Html.EJ().Sparkline("container").SparklineMouseLeave("sparklineMouseLeave"))
function mouseLeave(args) { } | Not applicable |
```

```
|Click| Event: click
@(Html.EJ().Sparkline("container").Click("click"))
function sparklineMouseClick(args) { } | Event: sparklineMouseClick
@Html.EJS().Sparkline("container").SparklineMouseClick("sparklineMouseClick").Render()
function sparklineMouseClick(args) { }
```

```
|doubleClick| Event: doubleClick
@(Html.EJ().Sparkline("container").DoubleClick("doubleClick"))
function doubleClick(args) { } | Not applicable |
```

```
|rightClick| Event: rightClick
@(Html.EJ().Sparkline("container").RightClick("rightClick"))
function rightClick(args) { } | Not applicable |
```

```
|axisRendering| Not applicable | Event: axisRendering
@Html.EJS().Sparkline("container").AxisRendering("axisRendering").Render()
function axisRendering(args) { }
```

```
|dataLabelRendering| Not applicable | Event: dataLabelRendering
@Html.EJS().Sparkline("container").DataLabelRendering("dataLabelRendering").Render()
function dataLabelRendering(args) { }
```

```
|markerRendering| Not applicable | Event: markerRendering
@Html.EJS().Sparkline("container").MarkerRendering("markerRendering").Render()
function markerRendering(args) { }
```

|pointRendering| Not applicable | **Event:** *pointRendering*<br/><br/>  
 @Html.EJS().Sparkline("container").PointRendering("pointRendering").Render()<br/><br/>function  
 on pointRendering(args) { } |

|resize| Not applicable | **Event:** *resize*<br/><br/>  
 @Html.EJS().Sparkline("container").Resize("resize").Render()<br/><br/>function resize(args) { } |

## SpeedDial

### Getting Started with ASP.NET MVC SpeedDial Control

This section briefly explains about how to include [ASP.NET MVC SpeedDial] control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

**~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC SpeedDial control

Now, add the Syncfusion ASP.NET MVC SpeedDial control in `~/Views/Home/Index.cshtml` page.

**CSHTML**

```
<div id="target" style="min-height:200px; position:relative; width:300px;
border:1px solid;">

@Html.EJS().SpeedDial("speeddial").Target("#target").Content("Edit").Items(V
iewBag.datasource).Render()
</div>
```

**DEFAULT.CS**

```
public ActionResult Demo()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Text="Cut"
 });
 items.Add(new SpeedDialItem
 {
 Text="Copy"
 });
}
```

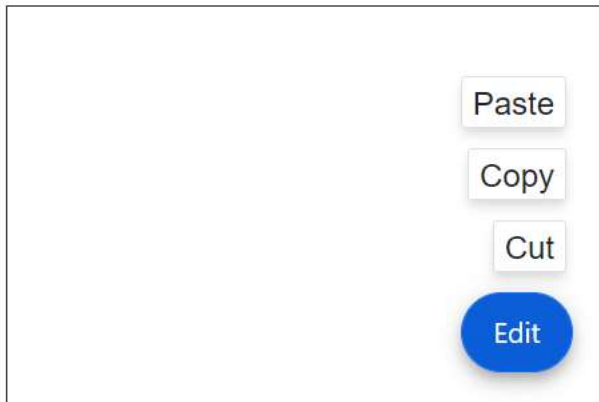
```

 items.Add(new SpeedDialItem
 {
 Text="Paste"
 });
 ViewBag.datasource = items;

 return View();
 }

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET Core SpeedDial control will be rendered in the default web browser.



### Positioning

The speed dial can be positioned using the [position](#) property. The speed dial is positioned based on the [target](#), if target is defined else positioned based on the browser viewport. The position values are TopLeft, TopCenter, TopRight, MiddleLeft, MiddleCenter, MiddleRight, BottomLeft, BottomCenter and BottomRight.

### CSHTML

```

<div id="target" style="min-height:200px; position:relative; width:300px;
border:1px solid;">

@Html.EJS().SpeedDial("element").Target("#target").Content("Edit").Position(
FabPosition.BottomLeft).Items(ViewBag.datasource).Render()
</div>

```

### POSITION.CS

```

public ActionResult Position()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut",
 Text="Cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy",
 Text="Copy"
 });
}

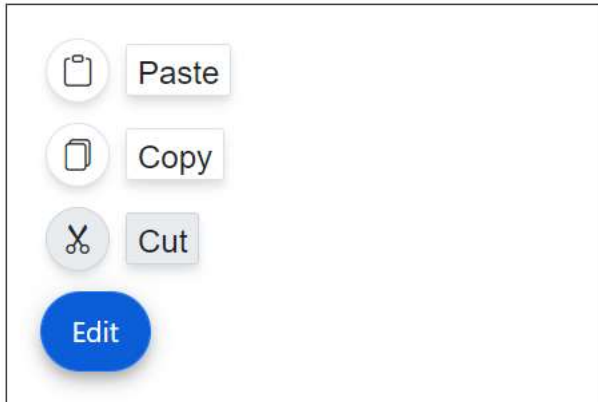
```

```

});
items.Add(new SpeedDialItem
{
 IconCss="e-icons e-paste",
 Text="Paste"
});
ViewBag.datasource = items;

return View();
}

```



### Linear and radial display modes

You can use the [Mode](#) property to either display the menu in linear order like a list or like a radial menu in radial (circular) direction.

### CSHTML

```

<div id="target" style="min-height:200px; position:relative; width:300px;
border:1px solid;">

@Html.EJS().SpeedDial("speeddial").Target("#target").Position(FabPosition.BottomLeft).Content("Edit").Mode(SpeedDialMode.Radial).OpenIconCss("e-icons e-edit").Items(ViewBag.datasource).Render()

@Html.EJS().SpeedDial("speeddial1").Target("#target").Position(FabPosition.BottomRight).Content("Edit").Mode(SpeedDialMode.Linear).OpenIconCss("e-icons e-edit").Items(ViewBag.datasourceLabel).Render()
</div>

```

### MODE.CS

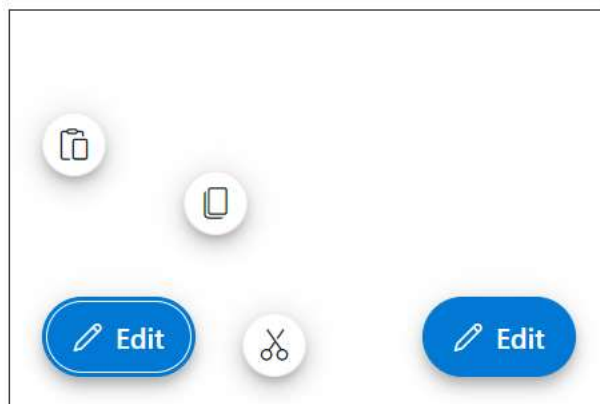
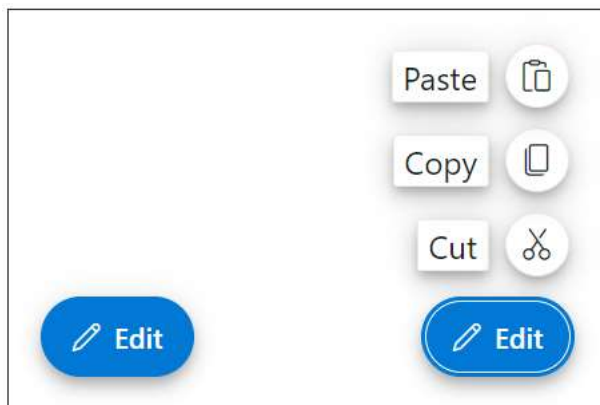
```

public ActionResult Mode()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 List<SpeedDialItem> items1 = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {

```

```
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 items1.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut",
 Text="Cut"
 });
 items1.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy",
 Text="Copy"
 });
 items1.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste",
 Text="Paste"
 });
 ViewBag.datasource = items;
 ViewBag.datasourceLabel = items1;

 return View();
}
```



### Clicked event

The speed dial control triggers the [Clicked](#) event when an action item is clicked.

You can use this event to perform the required action.

### CSHTML

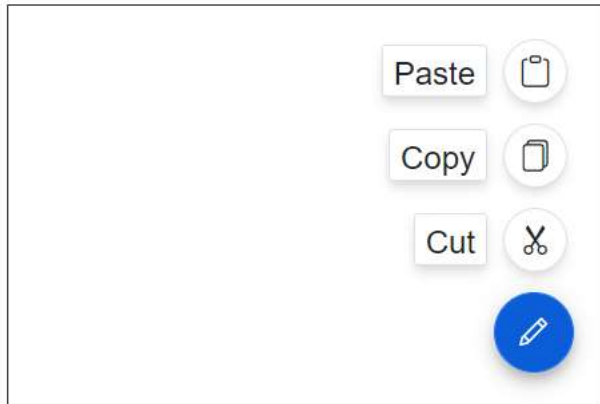
```
<div id="target" style="min-height:200px; position:relative; width:300px;
border:1px solid;">
 @Html.EJS().SpeedDial("speeddial").Target("#target").OpenIconCss("e-
icons e-
edit").Items(ViewBag.datasource).Clicked("function(args){clicked(args)}").Re
nder()
</div>
<script>
 function clicked(args) {
 alert(args.item.text + " is clicked");
 }
</script>
```

### ITEMCLICK.CS

```
public ActionResult ItemClick()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut",
 Text="Cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy",
 Text="Copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste",
 Text="Paste"
 });
 ViewBag.datasource = items;

 return View();
}
```





### Items in ASP.NET MVC Speed Dial Control

The action items in ASP.NET MVC Speed Dial can be added by using [Items](#) property.

| Fields                   | Type    | Description                                                                                        |
|--------------------------|---------|----------------------------------------------------------------------------------------------------|
| <a href="#">Text</a>     | string  | Defines the text content of SpeedDialItem.                                                         |
| <a href="#">IconCss</a>  | string  | Defines one or more CSS classes to include an icon or image in speed dial item.                    |
| <a href="#">Disabled</a> | boolean | Defines whether to enable or disable the SpeedDialItem.                                            |
| <a href="#">Id</a>       | string  | Defines a unique value for the SpeedDialItem which can be used to identify the item in event args. |
| <a href="#">Title</a>    | string  | Defines the title of SpeedDialItem to display tooltip.                                             |

### Icons in speeddial items

You can customize the icon and text of Speed Dial action items using [IconCss](#) and [Text](#) properties.

#### Icon only

You can show icon only in SpeedDial items by setting [IconCss](#) property. You can show tooltip on hover to show additional details to end-user by setting [Title](#) property.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).OpenIconCss("e-
icons e-edit").CloseIconCss("e-icons e-close").Render()
```

### ICON.CS

```
public ActionResult Icon()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Title="Cut",
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
```

```

 Title="Copy",
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 Title="Paste",
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}

```



### Text Only

You can show only text in Speed Dial items by setting [Text](#) property.

### CSHTML

```

@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Content("Edit")
.Render()

```

### TEXT.CS

```

public ActionResult Text()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Text="Cut"
 });
 items.Add(new SpeedDialItem
 {
 Text="Copy"
 });
 items.Add(new SpeedDialItem
 {
 Text="Paste"
 });
 ViewBag.datasource = items;

 return View();
}

```



### Icon with Text

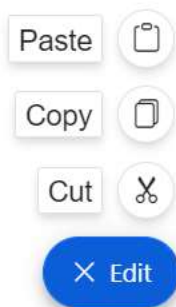
You can show icon along with text in Speed Dial items by setting [IconCss](#) and [Text](#) properties.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).OpenIconCss("e-
icons e-edit").CloseIconCss("e-icons e-close").Content("Edit").Render()
```

### TEXTICON.CS

```
public ActionResult TextIcon()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut",
 Text="Cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy",
 Text="Copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste",
 Text="Paste"
 });
 ViewBag.datasource = items;
}
```



### Disabled

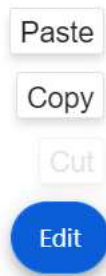
You can disable Speed Dial items by setting [Disabled](#) property as `true`.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Content("Edit")
.Render()
```

### DISABLED.CS

```
public ActionResult Icon()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Text="Cut",
 Disabled=true
 });
 items.Add(new SpeedDialItem
 {
 Text="Copy"
 });
 items.Add(new SpeedDialItem
 {
 Text="Paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Animation

The Speed Dial items can be animated during the opening and closing of the popup action items. You can customize the animation's Effect, Delay, and Duration by setting [Animation](#) property. By default, Speed Dial animates with a `Fade` effect and supports all [SpeedDialAnimationEffect](#) effects.

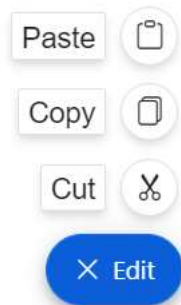
Below example demonstrates the Speed Dial items with applied Zoom effect.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Animation(new SpeedDialAnimationSettings
{
 Effect=SpeedDialAnimationEffect.Zoom}).Items(ViewBag.datasource).OpenIconCss
("e-icons e-edit").CloseIconCss("e-icons e-close").Content("Edit").Render()
```

### ANIMATION.CS

```
public ActionResult Animation()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Text = "Cut",
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 Text = "Copy",
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 Text = "Paste",
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Template

The Speed Dial supports to customize the action items and entire pop-up container by setting [ItemTemplate](#) and [PopupTemplate](#) property. For more details about templates, check out the link [here](#)

### Positions in ASP.NET MVC Speed Dial Control

The Speed dial control can be positioned anywhere on the [Target](#) using the [Position](#) property. If the [Target](#) is not defined, then Speed Dial is positioned based on the browser viewport.

The position values of Speed Dial are as follows:

- TopLeft
- TopCenter
- TopRight
- MiddleLeft
- MiddleCenter
- MiddleRight
- BottomLeft

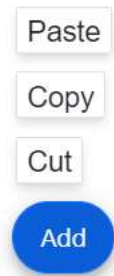
- BottomCenter
- BottomRight

**CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Position(FabPosition.BottomLeft).Content("Add").Render()
```

**POSITION.CS**

```
public ActionResult Position()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Text="Cut"
 });
 items.Add(new SpeedDialItem
 {
 Text="Copy"
 });
 items.Add(new SpeedDialItem
 {
 Text="Paste"
 });
 ViewBag.datasource = items;
 return View();
}
```

**Opens on hover**

You can open the Speed Dial action items on mouse hover by setting the [OpensOnHover](#) property as true.

**CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).OpensOnHover(true).OpenIconCss("e-icons e-edit").CloseIconCss("e-icons e-close").Render()
```

**HOVER.CS**

```
public ActionResult Hover()
{
}
```

```

List<SpeedDialItem> items = new List<SpeedDialItem>();
items.Add(new SpeedDialItem
{
 IconCss="e-icons e-cut"
});
items.Add(new SpeedDialItem
{
 IconCss="e-icons e-copy"
});
items.Add(new SpeedDialItem
{
 IconCss="e-icons e-paste"
});
ViewBag.datasource = items;
return View();
}

```



### Programmatically show/hide

You can open/close the Speed Dial action items programmatically using **show** and **hide** methods.

### CSHTML

```

@using Syncfusion.EJ2.Buttons
<div id="target" style="height:200px; position:relative; width:300px;
border:1px solid;">

@Html.EJS().Button("showbtn").Content("Show").Click("ShowItems()").Render()

@Html.EJS().Button("hidebtn").Content("Hide").Click("HideItems()").Render()

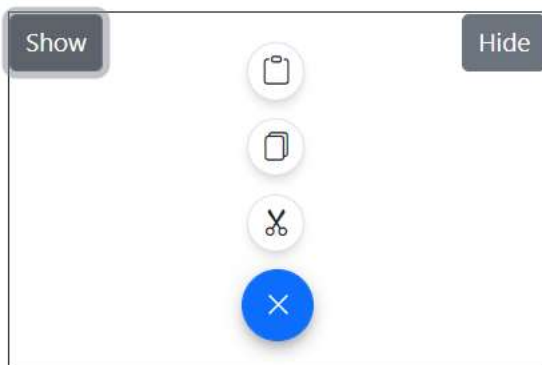
@Html.EJS().SpeedDial("speeddial").Target("#target").Items(ViewBag.datasource).Position(FabPosition.BottomCenter).OpenIconCss("e-icons e-edit").CloseIconCss("e-icons e-close").Render()
</div>
<style>
 #hidebtn{
 float:right;
 }
</style>
<script>
 function ShowItems()
 {

```

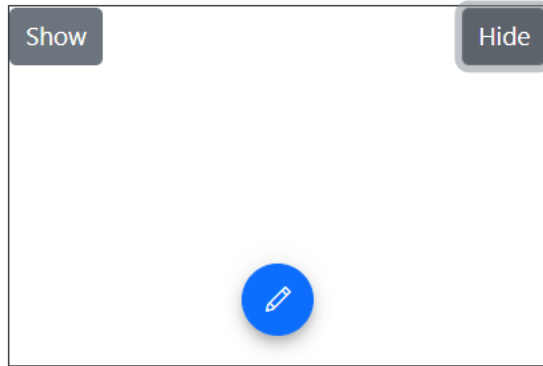
```
var speeddial =
document.getElementById('speeddial').ej2_instances[0];
speeddial.show();
}
function HideItems() {
var speeddial =
document.getElementById('speeddial').ej2_instances[0];
speeddial.hide();
}
</script>
```

## DISPLAY.CS

```
public ActionResult Display()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```







### Programmatically refresh the position

You can refresh the position of the Speed Dial using `refreshPosition` method when the `Targetposition` is changed.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
<div id="target" style="min-height:350px; position:relative; border:1px
solid;">

@Html.EJS().Button("refresh").Content("Refresh").Click("Refresh()").Render()

@Html.EJS().SpeedDial("speeddial").Target("#target").Items(ViewBag.datasourc
e).Position(FabPosition.MiddleRight).OpenIconCss("e-icons e-
edit").CloseIconCss("e-icons e-close").Render()
</div>
<script>
 function Refresh() {
 var speeddial =
document.getElementById('speeddial').ej2_instances[0];
 document.getElementById("target").style.minHeight = "300px";
 speeddial.refreshPosition();
 }
</script>
```

### REFRESH.CS

```
public ActionResult RefreshPosition()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
```

```
return View();
}
```

## Display Modes in ASP.NET MVC Speed Dial Control

The action items in ASP.NET MVC Speed Dial can be displayed in **Linear** and **Radial** display modes by setting **Mode** property.

### Linear display mode

In **Linear** display mode, Speed Dial action items are displayed in a list-like format either horizontally or vertically. By default, Speed Dial items are displayed in **Linear** mode.

### Direction

You can open the action items on the top, left, up, and down side of the Speed Dial button by setting **Direction** property. The default value is **Auto** where the action items are displayed based on the **Position** of the Speed Dial.

The **Linear** directions of Speed Dial are as follows:

- Left - Action items are displayed on the left side of the button.
- Right - Action items are displayed on the right side of the button.
- Up - Action items are displayed on the top of the button.
- Down - Action items are displayed on the bottom of the button.
- Auto - Action items display direction auto calculated based on **Position** of the Speed Dial. If Speed Dial is position at bottom right, then action items displayed at top.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Mode(SpeedDialMode.Linear).Direction(LinearDirection.Left).OpenIconCss("e-icons e-edit").Render()
```

### LINEARMODE.CS

```
public ActionResult LinearMode()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Radial display mode (Radial Menu)

In **Radial** mode, Speed Dial action items are displayed in a circular pattern like a radial menu. For more details about radial mode, check out the link [here](#).

### Radial Menu in ASP.NET MVC Speed Dial Control

The action items in ASP.NET MVC Speed Dial can be displayed in a circular pattern like a radial menu by setting **Mode** property. You can customize the **Direction**, **StartAngle**, **EndAngle** and **Offset** by setting [SpeedDialRadialSettings](#).

### Radial Menu direction

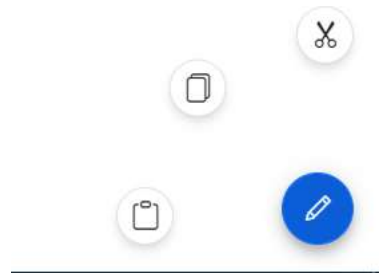
You can open the action items in either clockwise or anticlockwise by setting **Direction** property. The default value is **Auto** where the action items are displayed based on the **Position** property of the Speed Dial.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Mode(SpeedDialMode.Radial).RadialSettings
(new SpeedDialRadialSettings {
 Direction=RadialDirection.AntiClockwise}).Items(ViewBag.datasource).OpenIcon
Css("e-icons e-edit").Render()
```

### DIRECTION.CS

```
public ActionResult Direction()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Start and end angle

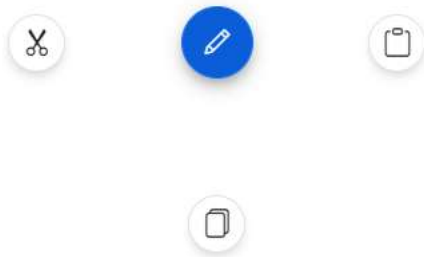
You can modify the start and end angle of action items by setting [StartAngle](#) and [EndAngle](#) properties. If the angle is not defined, the action items are displayed based on the **Position** property of the Speed Dial.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Mode(SpeedDialMode.Radial).RadialSettings
(new SpeedDialRadialSettings { Direction = RadialDirection.AntiClockwise,
StartAngle=180, EndAngle=360
}).Items(ViewBag.datasource).Position(FabPosition.MiddleCenter).OpenIconCss(
"e-icons e-edit").Render()
```

### ANGLES.CS

```
public ActionResult Angles()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Offset

You can modify the offset distance between action items and Speed Dial button using [Offset](#) property.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Mode(SpeedDialMode.Radial).RadialSettings
(new SpeedDialRadialSettings { Offset="80px"
}).Items(ViewBag.datasource).OpenIconCss("e-icons e-edit").Render()
```

### OFFSET.CS

```
public ActionResult Offset()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss = "e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Template in ASP.NET MVC SpeedDial Control

This section explains available templates in Speed Dial Control and its usage.

## Item template

You can use the [ItemTemplate](#) property to set a template content for the `SpeedDialItem`.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).ItemTemplate("#itemTemplate").Position(FabPosition.BottomRight).OpenIconCss("e-icons e-edit").Content("Edit").Render()
<script type="text/x-jsrender" id="itemTemplate">
 <div class="itemlist">

 ${text}
 </div>
</script>
<style>
 .e-speeddial-li .itemlist {
 display: inherit;
 width: 100%;
 border: 1px solid transparent;
 align-items: center;
 padding: 5px;
 border-radius: 500px;
 background-color: rgba(104, 99, 104, 0.1);
 box-shadow: 0 0 4px grey;
 }
</style>
```

### ITEMTEMPLATE.CS

```
public ActionResult ItemTemplate()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 Text="Cut",
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 Text="Copy",
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 Text="Paste",
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### Popup template

You can use the [PopupTemplate](#) property to set a template content for popup of SpeedDial control.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").PopupTemplate("#popupTemplate").Content("
FeedBack").Render()
<script type="text/x-jsrender" id="popupTemplate">
 <div class="speeddial-form">
 <p>Here you can customize your code.</p>
 </div>
</script>
<style>
 .speeddial-form {
 width: 200px;
 height: 80px;
 text-align: center;
 border-radius: 15px;
 box-shadow: rgb(0 0 0 / 10%) 0px 10px 15px -3px, rgb(0 0 0 / 5%) 0px
4px 6px -2px;
 background: #f5f5f5;
 padding: 15px;
 }
</style>
```

Here you can  
customize your code.

### Styles in ASP.NET MVC SpeedDial Control

This section briefs different ways to style SpeedDial Control.

### SpeedDial button

You can customize the icon and text of ASP.NET MVC SpeedDial using [OpenIconCss](#), [CloseIconCss](#) and [Content](#) properties.

#### *SpeedDial with Icon*

You can use the [OpenIconCss](#) and [CloseIconCss](#) property to show icons in speed dial button. You can also show tooltip on hover to show additional details to end-user by setting `title` attribute.

#### **CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).OpenIconCss("e-icons e-edit").CloseIconCss("e-icons e-close").Render()
```

#### **ICON.CS**

```
public ActionResult Icon()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



#### *SpeedDial with Text*

You can show text only in SpeedDial button by setting [Content](#) property without setting icon properties.

#### **CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Content("Edit").Render()
```



**TEXT.CS**

```

public ActionResult Text()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}

```

*SpeedDial with Icon and Text*

You can show the icon and text of ASP.NET MVC Speed Dial Button using [OpenIconCss](#), [CloseIconCss](#) and [Content](#) properties together.

**CSHTML**

```

@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).OpenIconCss("e-
icons e-edit").CloseIconCss("e-icons e-close").Content("Edit").Render()

```

**ICONTEXT.CS**

```

public ActionResult IconText()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
}

```

```
items.Add(new SpeedDialItem
{
 IconCss="e-icons e-paste"
});
ViewBag.datasource = items;
return View();
}
```



### Disabled

You can enable or disable the Speed Dial Control using [Disabled](#) property.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Disabled(true).
Content("Edit").Render()
```

### DISABLED.CS

```
public ActionResult Disabled()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### CssClass

The ASP.NET MVC Speed Dial supports the following predefined styles that can be defined using the [CssClass](#) property. You can customize by setting the `CssClass` property with the below defined class.

| CssClass  | Description                                             |
|-----------|---------------------------------------------------------|
| -----     | -----                                                   |
| e-primary | Used to represent a primary action.                     |
| e-outline | Used to represent an appearance of button with outline. |
| e-info    | Used to represent an informative action.                |
| e-success | Used to represent a positive action.                    |
| e-warning | Used to represent an action with caution.               |
| e-danger  | Used to represent a negative action.                    |

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).CssClass("e-
warning").Content("Edit").Render()
```

### CSSCLASS.CS

```
public ActionResult CssClass()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



## Visible

You can set the Speed Dial Control to visible/hidden state using **Visible** property.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Visible(false).
Content("Edit").Render()
```

### VISIBLE.CS

```
public ActionResult Visible()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```

## Tooltip

You can show tooltip on hover to show additional details to end-user by setting **Title** property to speed dial items.

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Content("Edit")
.Render()
```

### TOOLTIP.CS

```
public ActionResult Tooltip()
{
```

```

List<SpeedDialItem> items = new List<SpeedDialItem>();
items.Add(new SpeedDialItem
{
 Title="Cut",
 IconCss="e-icons e-cut"
});
items.Add(new SpeedDialItem
{
 Title="Copy",
 IconCss="e-icons e-copy"
});
items.Add(new SpeedDialItem
{
 Title="Paste",
 IconCss="e-icons e-paste"
});
ViewBag.datasource = items;
return View();
}

```

### Opens on hover

You can use [OpensOnHover](#) property to open actions items on hover itself. By default action items displayed only when clicking the speed dial button.

### CSHTML

```

@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).OpensOnHover(true).Content("Edit").Render()

```

### HOVER.CS

```

public ActionResult Hover()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}

```



### Modal in ASP.NET MVC SpeedDial Control

You can use the [Modal](#) property to set the Speed Dial as modal which adds an overlay to prevent the background interaction.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").Items(ViewBag.datasource).Modal(true).Position(FabPosition.BottomRight).OpenIconCss("e-icons e-edit").Render()
```

#### MODAL.CS

```
public ActionResult Modal()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



## Events in ASP.NET MVC SpeedDial Control

This section describes the Speed Dial events that will be triggered when appropriate actions are performed. The following events are available in the Speed Dial Control.

### Clicked event

The speed dial control triggers the [Clicked](#) event when an action item is clicked. You can use this event to perform the required action.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").Clicked("ClickedEvent").Items(ViewBag.datasource).Render()
<script>
 function ClickedEvent(args) {
 //Your required action here
 }
</script>
```

#### CLICKEDEVENT.CS

```
public ActionResult ItemClick()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```

### Created

The speed dial control triggers the [Created](#) event when SpeedDial control rendering is completed.

#### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").Created("CreatedEvent").Items(ViewBag.datasource).Render()
<script>
 function CreatedEvent() {
 //Your required action here
 }
</script>
```

**CREATEDEVENT.CS**

```
public ActionResult ItemClick()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```

**Before open**

The speed dial control triggers the [BeforeOpen](#) event before the SpeedDial popup is opened.

**CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").BeforeOpen("BeforeOpenEvent").Items(ViewBag.datasource).Render()
<script>
 function BeforeOpenEvent() {
 //Your required action here
 }
</script>
```

**BEFOREOPENEVENT.CS**

```
public ActionResult BeforeOpenEvent()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```



### On open

The speed dial control triggers the [OnOpen](#) event when SpeedDial popup is opened.

#### **CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").OnOpen("OnOpenEvent").Items(ViewBag.datasource).Render()
<script>
 function OnOpenEvent() {
 //Your required action here
 }
</script>
```

#### **ONOPENEVENT.CS**

```
public ActionResult OnOpenEvent()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });

 ViewBag.datasource = items;
 return View();
}
```

### Before close

The speed dial control triggers the [BeforeClose](#) event before the SpeedDial popup is closed.

#### **CSHTML**

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").BeforeClose("BeforeCloseEvent").Items(ViewBag.datasource).Render()
<script>
 function BeforeCloseEvent() {
 //Your required action here
 }
</script>
```

#### **BEFORECLOSEEVENT.CS**

```
public ActionResult BeforeCloseEvent()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
```

```

 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
 }

```

### On close

The speed dial control triggers the [OnClose](#) event when SpeedDial popup is closed.

### CSHTML

```

@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").OnClose("OnCloseEvent").Items(ViewBag.datasource).Render()
<script>
 function OnCloseEvent() {
 //Your required action here
 }
</script>

```

### ONCLOSEEVENT.CS

```

public ActionResult OnCloseEvent()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}

```

### Before item render

The speed dial control triggers the [BeforeItemRender](#) event for each `SpeedDialItem` once its rendered..

### CSHTML

```
@using Syncfusion.EJ2.Buttons
@Html.EJS().SpeedDial("speeddial").OpenIconCss("e-icons e-
edit").BeforeItemRender("BeforeItemRenderEvent").Items(ViewBag.datasource).R
ender()
<script>
 function BeforeItemRenderEvent() {
 //Your required action here
 }
</script>
```

### ITEMRENDEREVENT.CS

```
public ActionResult ItemRenderEvent()
{
 List<SpeedDialItem> items = new List<SpeedDialItem>();
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-cut"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-copy"
 });
 items.Add(new SpeedDialItem
 {
 IconCss="e-icons e-paste"
 });
 ViewBag.datasource = items;
 return View();
}
```

### Accessibility in ASP.NET MVC SpeedDial Control

Accessibility is achieved in the Speed Dial control through WAI-ARIA standard and keyboard navigations. The Speed Dial control can be effectively accessed through assistive technologies such as screen readers.

#### Keyboard interaction

The Speed Dial control is interactive with below keyboard shortcuts.

| Keyboard shortcuts | Actions |

|-----|-----|

| Enter | Open/close the menu. If a SpeedDial item is focused, should triggers the clicked event for the item. |

| ArrowUp | Navigates up or to the previous menu item. |

| ArrowLeft | Navigates left or to the previous menu item. |

| ArrowDown- | Navigates down or to the previous menu item. |

| ArrowRight | Navigates right or to the previous menu item. |

| Home | Navigates to the first menu item. |

| End | Navigates to the last menu item. |

| Esc | Closes the menu. |

### ARIA attributes

The following ARIA attributes are applicable for SpeedDial Control based on its state.

| Properties | Functionality |

| ----- | ----- |

| role | This attribute is added to the input element to describe the actual role. |

| aria-label | Attribute provides the text label with some default description for the SpeedDial and its items. |

| aria-expanded | It indicates whether the SpeedDial current state is expanded or collapsed. |

| aria-haspopup | It indicates whether the SpeedDial has popup items or not. |

| aria-controls | Attribute is set to the SpeedDial button and it points to the corresponding content. |

| aria-disabled | It indicates the disabled state of the SpeedDial and its items. |

## Spinner

### Getting Started with ASP.NET MVC Spinner Control

This section briefly explains about how to include ASP.NET MVC Spinner control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

#### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
`
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
`
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Spinner Control

Initialize the Spinner using `createSpinner` method and show/hide the spinner using `showSpinner` and `hideSpinner` methods accordingly. Set the target to the spinner to render it based on specific target.

Now, add the `Spinner` using Syncfusion Essential JavaScript library in `~/Views/Home/Index.cshtml` page.

```
`cshtml
....
<div id="container" style="position:center; height: 300px;">
```

<p>If you click the <b>Show Spinner</b> button you can create the <b>ASP.NET MVC Spinner component</b> </p>

```
<button id="showSpinnerButton">Show Spinner</button>
```

```
</div>
```

```
<script>
```

```
var spinTarget = document.getElementById('container');
```

```
ej.popups.createSpinner({
```

```
target: spinTarget
```

```
});
```

```
// Attach a click event handler to the "Show Spinner" button
```

```
document.getElementById('showSpinnerButton').addEventListener('click', function () {
```

```
ej.popups.showSpinner(spinTarget);
```

```
setInterval(function () {
```

```
//hideSpinner() method is used to hide the Spinner
```

```
ej.popups.hideSpinner(spinTarget);
```

```
, 5000);
```

```
});
```

```
</script>
```

```
,
```

- Show and hide this spinner by using `showSpinner` and `hideSpinner` methods for loading in your page.

### Set the template to the Spinner

You can use custom templates on the Spinner instead of the default Spinner by specifying the template in the `setSpinner` method.

The following steps explain you on how to define template for Spinner.

- Pass your custom template to the `setSpinner` method like as below.

```
`typescript
```

```
// Specify the template content to be displayed in the Spinner
```

```
setSpinner({ template: '<div style="width:100%;height:100%" class="custom-rolling"><div></div></div>' });
```

```
,
```

**Note:** You should set the template to the Spinner before creating the respective Essential JS 2 component. Also, until we replace `setSpinner` template, the further Essential JS 2 component rendering is created with given template only.

- Now, render the Essential JS 2 component. It's render the Spinner with the template specified in the `setSpinner` method.

**Note:** In the below sample, we have rendered the Grid component with custom Spinner using `setSpinner` method. You have to define the styles for the template in style section.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Grid
<div class="col-lg-12 control-section">
 <div class="e-sample-resize-container">

@Html.EJ().Grid("ej2Grid1").DataSource(ViewBag.datasources).Created("grid1Created").AllowPaging(true).Columns(new List<GridColumn> {
 new GridColumn { Field = "OrderID", HeaderText = "Order ID",
Width = "120" },
 new GridColumn { Field = "CustomerID", HeaderText = "Customer
Name", Width = "150" },
 new GridColumn { Field = "OrderDate", HeaderText = "Order Date",
Width = "130", Format="yMd" },
 new GridColumn { Field = "Freight", HeaderText = "Freight",
Width = "120" },
 new GridColumn { Field = "ShippedDate", HeaderText = "Shipped
Date", Width = "140", Format="yMd" },
 new GridColumn { Field = "ShipCountry", HeaderText = "Ship
Country", Width = "150" },
}).Render()

@Html.EJ().Grid("ej2Grid2").DataSource(ViewBag.datasources).Created("grid2Created").AllowPaging(true).Columns(new List<GridColumn> {
 new GridColumn { Field = "OrderID", HeaderText = "Order ID",
Width = "120" },
 new GridColumn { Field = "CustomerID", HeaderText = "Customer
Name", Width = "150" },
 new GridColumn { Field = "OrderDate", HeaderText = "Order Date",
Width = "130", Format="yMd" },
 new GridColumn { Field = "Freight", HeaderText = "Freight",
Width = "120" },
 new GridColumn { Field = "ShippedDate", HeaderText = "Shipped
Date", Width = "140", Format="yMd" },
 new GridColumn { Field = "ShipCountry", HeaderText = "Ship
Country", Width = "150" },
}).Render()
 </div>
</div>
<style>
#container {
```

```

 visibility: hidden;
 }
 #loader {
 color: #008cff;
 font-family: 'Helvetica Neue','calibiri';
 font-size: 14px;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .wrap {
 margin: 0 auto;
 width: 240px;
 }
 h5 {
 font-weight: bold;
 text-align: center;
 }
 @@keyframes custom-rolling {
 0% {
 -webkit-transform: translate(-50%, -50%) rotate(0deg);
 transform: translate(-50%, -50%) rotate(0deg);
 }
 100% {
 -webkit-transform: translate(-50%, -50%) rotate(360deg);
 transform: translate(-50%, -50%) rotate(360deg);
 }
 }
 @@-webkit-keyframes custom-rolling {
 0% {
 -webkit-transform: translate(-50%, -50%) rotate(0deg);
 transform: translate(-50%, -50%) rotate(0deg);
 }
 100% {
 -webkit-transform: translate(-50%, -50%) rotate(360deg);
 transform: translate(-50%, -50%) rotate(360deg);
 }
 }
 .custom-rolling {
 position: relative;
 }
 .custom-rolling div,
 .custom-rolling div:after {
 border: 16px solid #51CACC;
 border-radius: 50%;
 border-top-color: transparent;
 height: 160px;
 position: absolute;
 width: 160px;
 }
 .custom-rolling div {
 -webkit-animation: custom-rolling 1.3s linear infinite;
 animation: custom-rolling 1.3s linear infinite;
 top: 100px;
 left: 100px;
 }

```



```

 }
 .custom-rolling div:after {
 -ms-transform: rotate(90deg);
 -webkit-transform: rotate(90deg);
 transform: rotate(90deg);
 }
 .custom-rolling {
 -webkit-transform: translate(-31px, -31px) scale(0.31)
translate(31px, 31px);
 height: 62px !important;
 transform: translate(-31px, -31px) scale(0.31) translate(31px,
31px);
 width: 62px !important;
 }
</style>
<script type="text/javascript">
 function grid1Created() {
 var grid1 = document.getElementById("ej2Grid1").ej2_instances[0];
 grid1.hideSpinner = () => true;
 ej.popups.setSpinner({ template: '<div
style="width:100%;height:100%" class="custom-rolling"><div></div></div>' });
 }
 function grid2Created() {
 var grid2 = document.getElementById("ej2Grid2").ej2_instances[0];
 grid2.hideSpinner = () => true;
 }
</script>

```

## TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Grid;
namespace WebApplication1.Controllers
{
 public partial class SpinnerController : Controller
 {
 public static List<OrderDetails> order = new List<OrderDetails>();
 public IActionResult template()
 {
 if (order.Count() == 0)
 BindDataSource();
 ViewBag.datasource = order;
 return View();
 }
 public void BindDataSource()
 {
 int code = 10000;
 for (int i = 1; i < 10; i++)
 {
 order.Add(new OrderDetails(code + 1, "ALFKI", i + 0, 2.3 *
i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro", "Denmark",
new DateTime(1996, 7, 16), "Kirchgasse 6"));
 }
 }
 }
}

```

```

 order.Add(new OrderDetails(code + 2, "ANATR", i + 2, 3.3 *
i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha", "Brazil",
new DateTime(1996, 9, 11), "Avda. Azteca 123"));
 order.Add(new OrderDetails(code + 3, "ANTON", i + 1, 4.3 *
i, true, new DateTime(1957, 11, 30), "Cholchester", "Frankenversand",
"Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave. Bolívar #65-98
Llano Largo"));
 order.Add(new OrderDetails(code + 4, "BLONP", i + 3, 5.3 *
i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel",
"Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
 order.Add(new OrderDetails(code + 5, "BOLID", i + 4, 6.3 *
i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes",
"Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
 code += 5;
 }
}
[Serializable]
public class OrderDetails
{
 public OrderDetails()
 {
 }
 public OrderDetails(int OrderID, string CustomerID, int
EmployeeID, double Freight, bool Verified, DateTime OrderDate, string
ShipCity, string ShipName, string ShipCountry, DateTime ShippedDate, string
ShipAddress)
 {
 this.OrderID = OrderID;
 this.CustomerID = CustomerID;
 this.EmployeeID = EmployeeID;
 this.Freight = Freight;
 this.ShipCity = ShipCity;
 this.Verified = Verified;
 this.OrderDate = OrderDate;
 this.ShipName = ShipName;
 this.ShipCountry = ShipCountry;
 this.ShippedDate = ShippedDate;
 this.ShipAddress = ShipAddress;
 }
 public int? OrderID { get; set; }
 public string CustomerID { get; set; }
 public int? EmployeeID { get; set; }
 public double? Freight { get; set; }
 public string ShipCity { get; set; }
 public bool Verified { get; set; }
 public DateTime OrderDate { get; set; }
 public string ShipName { get; set; }
 public string ShipCountry { get; set; }
 public DateTime ShippedDate { get; set; }
 public string ShipAddress { get; set; }
}
}
}

```

## Change the type of the Spinner

By default, the Spinner is loaded in the applicable Essential JS 2 component based on the theme imported into the page. Based on the theme, the type is set to the Spinner.

The available types are:

- Material
- Fabric
- Bootstrap

You can change the Essential JS 2 component spinner type by passing the type of the spinner as parameter to the `setSpinner` method like as below.

```
`typescript
// Specify the type of the Spinner to be displayed
setSpinner({ type: 'Bootstrap'});
`
```

**Note:** After Essential JS 2 component creation only, you can change the Essential JS 2 component spinner type.

### CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Grid;
<div class="col-lg-12 control-section">
 <div class="e-sample-resize-container">

@Html.EJ().Grid("ej2Grid").DataSource(ViewBag.datasources).Created("gridCreated").AllowPaging(true).Columns(new List<GridColumn> {
 new GridColumn { Field = "OrderID", HeaderText = "Order ID",
Width = "120" },
 new GridColumn { Field = "CustomerID", HeaderText = "Customer
Name", Width = "150" },
 new GridColumn { Field = "OrderDate", HeaderText = "Order Date",
Width = "130", Format="yMd" },
 new GridColumn { Field = "Freight", HeaderText = "Freight",
Width = "120" },
 new GridColumn { Field = "ShippedDate", HeaderText = "Shipped
Date", Width = "140", Format="yMd" },
 new GridColumn { Field = "ShipCountry", HeaderText = "Ship
Country", Width = "150" },
}).Render()
 </div>
</div>
<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 font-family: 'Helvetica Neue','calibiri';
 font-size: 14px;
 height: 40px;
```

```

 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .wrap {
 margin: 0 auto;
 width: 240px;
 }
 h5 {
 font-weight: bold;
 text-align: center;
 }
</style>
<script type="text/javascript">
 function gridCreated() {
 var grid = document.getElementById("ej2Grid").ej2_instances[0];
 grid.hideSpinner = () => true;
 ej.popups.setSpinner({ type: 'Bootstrap' });
 }
</script>

```

## TYPE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Grid;
namespace WebApplication1.Controllers
{
 public partial class SpinnerController : Controller
 {
 public static List<OrderDetailList> orders = new
List<OrderDetailList>();
 public IActionResult type()
 {
 if (orders.Count() == 0)
 BindDataSources();
 ViewBag.datasource = order;
 return View();
 }
 public void BindDataSources()
 {
 int code = 10000;
 for (int i = 1; i < 10; i++)
 {
 orders.Add(new OrderDetailList(code + 1, "ALFKI", i + 0, 2.3
* i, false, new DateTime(1991, 05, 15), "Berlin", "Simons bistro",
"Denmark", new DateTime(1996, 7, 16), "Kirchgasse 6"));
 orders.Add(new OrderDetailList(code + 2, "ANATR", i + 2, 3.3
* i, true, new DateTime(1990, 04, 04), "Madrid", "Queen Cozinha", "Brazil",
new DateTime(1996, 9, 11), "Avda. Azteca 123"));
 orders.Add(new OrderDetailList(code + 3, "ANTON", i + 1, 4.3
* i, true, new DateTime(1957, 11, 30), "Cholchester", "Frankenversand",

```

```

"Germany", new DateTime(1996, 10, 7), "Carrera 52 con Ave. Bolívar #65-98
Llano Largo"));
 orders.Add(new OrderDetailList(code + 4, "BLONP", i + 3, 5.3
* i, false, new DateTime(1930, 10, 22), "Marseille", "Ernst Handel",
"Austria", new DateTime(1996, 12, 30), "Magazinweg 7"));
 orders.Add(new OrderDetailList(code + 5, "BOLID", i + 4, 6.3
* i, true, new DateTime(1953, 02, 18), "Tsawassen", "Hanari Carnes",
"Switzerland", new DateTime(1997, 12, 3), "1029 - 12th Ave. S.));
 code += 5;
 }
}
[Serializable]
public class OrderDetailList
{
 public OrderDetailList()
 {
 }
 public OrderDetailList(int OrderID, string CustomerId, int
EmployeeId, double Freight, bool Verified, DateTime OrderDate, string
ShipCity, string ShipName, string ShipCountry, DateTime ShippedDate, string
ShipAddress)
 {
 this.OrderID = OrderID;
 this.CustomerID = CustomerId;
 this.EmployeeID = EmployeeId;
 this.Freight = Freight;
 this.ShipCity = ShipCity;
 this.Verified = Verified;
 this.OrderDate = OrderDate;
 this.ShipName = ShipName;
 this.ShipCountry = ShipCountry;
 this.ShippedDate = ShippedDate;
 this.ShipAddress = ShipAddress;
 }
 public int? OrderID { get; set; }
 public string CustomerID { get; set; }
 public int? EmployeeID { get; set; }
 public double? Freight { get; set; }
 public string ShipCity { get; set; }
 public bool Verified { get; set; }
 public DateTime OrderDate { get; set; }
 public string ShipName { get; set; }
 public string ShipCountry { get; set; }
 public DateTime ShippedDate { get; set; }
 public string ShipAddress { get; set; }
}
}
}

```

### CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the spinner

Use the following CSS to customize the spinner stroke color.

### *Material theme*

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-material {
stroke: green;
}
,
```

### *Fabric theme*

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-fabric {
stroke: green;
}
,
```

### *Bootstrap theme*

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap {
fill: green;
stroke: green;
}
,
```

### *Bootstrap4 theme*

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-bootstrap4 {
stroke: green;
}
,
```

### *High Contrast theme*

`CSS

```
.e-spinner-pane .e-spinner-inner .e-spin-high-contrast .e-path-arc {
stroke: green;
}
,
```

## How To

### Real Time Example Using Spinner

In the following sample, you can see the Spinner when expanding the collapsed swimlane rows in Kanban control. The Spinner indicates the loading data while fetching the data.

**CSHTML**

```

@using Syncfusion.EJ2.Popups
<div id="container" style="position: relative; height: 300px;">

@Html.EJS().Kanban("kanban").KeyField("Status").ActionBegin("actionBegin").D
ataBound("onDataBound").DataSource((IEnumerable<object>
) ViewBag.data).Columns(col =>
{
 col.HeaderText("To Do").KeyField("Open").Add();
 col.HeaderText("In Progress").KeyField("InProgress").Add();
 col.HeaderText("Testing").KeyField("Testing").Add();
 col.HeaderText("Done").KeyField("Close").Add();
}).CardSettings(card =>
{
 card.ContentField("Summary").HeaderField("Id");
}).SwimlaneSettings(swim =>
{
 swim.KeyField("Assignee");
}).Render()
</div>
<script>
var spinTarget = document.getElementById('container');
ej.popups.createSpinner({
 target: spinTarget
});
var flag = true;
function onDataBound() {
 if (flag) {
 var ele = document.querySelectorAll(".e-icons.e-swimlane-row-
expand");
 for (var i = 0; i < ele.length; i++) {
 ele[i].click();
 }
 flag = false;
 }
}
function actionBegin(args) {
 if (args.requestType === "rowExpandCollapse") {
 ej.popups.showSpinner(spinTarget);
 setInterval(function () {
 //hideSpinner() method is used to hide the Spinner
 ej.popups.hideSpinner(spinTarget);
 }, 5000);
 }
}
</script>

```

**CONTROLLER.CS**

```

public class HomeController : Controller
{

 public ActionResult Index()
 {
 ViewBag.data = new KanbanDataModels().KanbanTasks();
 }
}

```

```

 return View();
 }
 public class KanbanDataModels
 {
 public string Id { get; set; }
 public string Title { get; set; }
 public string Status { get; set; }
 public string Summary { get; set; }
 public string Type { get; set; }
 public string Priority { get; set; }
 public string Tags { get; set; }
 public Double Estimate { get; set; }
 public string Assignee { get; set; }
 public int RankId { get; set; }
 public string Color { get; set; }
 public List<KanbanDataModels> KanbanTasks()
 {
 List<KanbanDataModels> TaskDetails = new
List<KanbanDataModels>();
 TaskDetails.Add(new KanbanDataModels { Id = "Task 1", Title
= "Task - 29001", Status = "Open", Summary = "Analyze the new requirements
gathered from the customer.", Type = "Story", Priority = "Low", Tags =
"Analyze, Customer", Estimate = 3.5, Assignee = "Nancy Davloio", RankId = 1,
Color = "#8b447a" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 2", Title
= "Task - 29002", Status = "InProgress", Summary = "Improve application
performance", Type = "Improvement", Priority = "Normal", Tags =
"Improvement", Estimate = 6, Assignee = "Andrew Fuller", RankId = 1, Color =
"#7d7297" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 3", Title
= "Task - 29003", Status = "Open", Summary = "Arrange a web meeting with
the customer to get new requirements.", Type = "Others", Priority =
"Critical", Tags = "Meeting", Estimate = 5.5, Assignee = "Janet Leverling",
RankId = 2, Color = "#27AE60" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 4", Title
= "Task - 29004", Status = "InProgress", Summary = "Fix the issues reported
in the IE browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"IE", Estimate = 2.5, Assignee = "Janet Leverling", RankId = 2, Color =
"#cc0000" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 5", Title
= "Task - 29005", Status = "Review", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Low", Tags = "Customer", Estimate
= 3.5, Assignee = "Steven walker", RankId = 1, Color = "#cc0000" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 6", Title
= "Task - 29007", Status = "Validate", Summary = "Validate new
requirements", Type = "Improvement", Priority = "Low", Tags = "Validation",
Estimate = 1.5, Assignee = "Robert King", RankId = 1, Color = "#7d7297" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 7", Title
= "Task - 29009", Status = "Review", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix, Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2, Color
= "#cc0000" });
 TaskDetails.Add(new KanbanDataModels { Id = "Task 8", Title
= "Task - 29010", Status = "Close", Summary = "Test the application in the
IE browser.", Type = "Story", Priority = "Low", Tags = "Review, IE", Estimate
= 5.5, Assignee = "Margaret hamilt", RankId = 3, Color = "#8b447a" });
 }
 }
}

```



```

 TaskDetails.Add(new KanbanDataModels { Id = "Task 9", Title
= "Task - 29011", Status = "Validate", Summary = "Validate the issues
reported by the customer.", Type = "Story", Priority = "High", Tags =
"Validation,Fix", Estimate = 1, Assignee = "Steven walker", RankId = 1,
Color = "#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 10", Title
= "Task - 29015", Status = "Open", Summary = "Show the retrieved data from
the server in grid control.", Type = "Story", Priority = "High", Tags =
"Database,SQL", Estimate = 5.5, Assignee = "Margaret hamilt", RankId = 4,
Color = "#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 11", Title
= "Task - 29016", Status = "InProgress", Summary = "Fix cannot open user's
default database SQL error.", Priority = "Critical", Type = "Bug", Tags =
"Database,Sql2008", Estimate = 2.5, Assignee = "Janet Leverling", RankId =
4, Color = "#cc0000" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 12", Title
= "Task - 29017", Status = "Review", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4, Color = "#8b447a"
});

 TaskDetails.Add(new KanbanDataModels { Id = "Task 13", Title
= "Task - 29018", Status = "Close", Summary = "Analyze SQL server 2008
connection.", Type = "Story", Priority = "Release Breaker", Tags =
"Grid,Sql", Estimate = 2, Assignee = "Andrew Fuller", RankId = 4, Color =
"#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 14", Title
= "Task - 29019", Status = "Validate", Summary = "Validate databinding
issues.", Type = "Story", Priority = "Low", Tags = "Validation", Estimate =
1.5, Assignee = "Margaret hamilt", RankId = 1, Color = "#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 15", Title
= "Task - 29020", Status = "Close", Summary = "Analyze grid control.", Type
= "Story", Priority = "High", Tags = "Analyze", Estimate = 2.5, Assignee =
"Margaret hamilt", RankId = 5, Color = "#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 16", Title
= "Task - 29021", Status = "Close", Summary = "Stored procedure for initial
data binding of the grid.", Type = "Others", Priority = "Release Breaker",
Tags = "Databinding", Estimate = 1.5, Assignee = "Steven walker", RankId =
6, Color = "#27AE60" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 17", Title
= "Task - 29022", Status = "Close", Summary = "Analyze stored procedures.",
Type = "Story", Priority = "Release Breaker", Tags = "Procedures", Estimate
= 5.5, Assignee = "Janet Leverling", RankId = 7, Color = "#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 18", Title
= "Task - 29023", Status = "Validate", Summary = "Validate editing
issues.", Type = "Story", Priority = "Critical", Tags = "Editing", Estimate
= 1, Assignee = "Nancy Davloio", RankId = 1, Color = "#8b447a" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 19", Title
= "Task - 29024", Status = "Review", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5, Color = "#8b447a"
});

 TaskDetails.Add(new KanbanDataModels { Id = "Task 20", Title
= "Task - 29025", Status = "Open", Summary = "Enhance editing
functionality.", Type = "Improvement", Priority = "Low", Tags = "Editing",
Estimate = 3.5, Assignee = "Andrew Fuller", RankId = 5, Color = "#7d7297"
});

```

```

 TaskDetails.Add(new KanbanDataModels { Id = "Task 21", Title
= "Task - 29026", Status = "InProgress", Summary = "Improve the performance
of the editing functionality.", Type = "Epic", Priority = "High", Tags =
"Performance", Estimate = 6, Assignee = "Nancy Davloio", RankId = 5, Color =
"#6d7492" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 22", Title
= "Task - 29027", Status = "Open", Summary = "Arrange web meeting with the
customer to show editing demo.", Type = "Others", Priority = "High", Tags =
"Meeting,Editing", Estimate = 5.5, Assignee = "Steven walker", RankId = 6,
Color = "#27AE60" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 23", Title
= "Task - 29029", Status = "Review", Summary = "Fix the editing issues
reported by the customer.", Type = "Bug", Priority = "Low", Tags =
"Editing,Fix", Estimate = 3.5, Assignee = "Janet Leverling", RankId = 6,
Color = "#cc0000" });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 24", Title
= "Task - 29030", Status = "Testing", Summary = "Fix the issues reported by
the customer.", Type = "Bug", Priority = "Critical", Tags = "Customer",
Estimate = 3.5, Assignee = "Steven walker", RankId = 1 });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 25", Title
= "Task - 29031", Status = "Testing", Summary = "Fix the issues reported in
Safari browser.", Type = "Bug", Priority = "Release Breaker", Tags =
"Fix,Safari", Estimate = 1.5, Assignee = "Nancy Davloio", RankId = 2 });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 26", Title
= "Task - 29032", Status = "Testing", Summary = "Check Login page
validation.", Type = "Story", Priority = "Release Breaker", Tags =
"Testing", Estimate = 0.5, Assignee = "Michael Suyama", RankId = 3 });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 27", Title
= "Task - 29033", Status = "Testing", Summary = "Fix the issues reported in
data binding.", Type = "Story", Priority = "Normal", Tags = "Databinding",
Estimate = 3.5, Assignee = "Janet Leverling", RankId = 4 });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 28", Title
= "Task - 29034", Status = "Testing", Summary = "Test editing
functionality.", Type = "Story", Priority = "Normal", Tags = "Editing,Test",
Estimate = 0.5, Assignee = "Nancy Davloio", RankId = 5 });

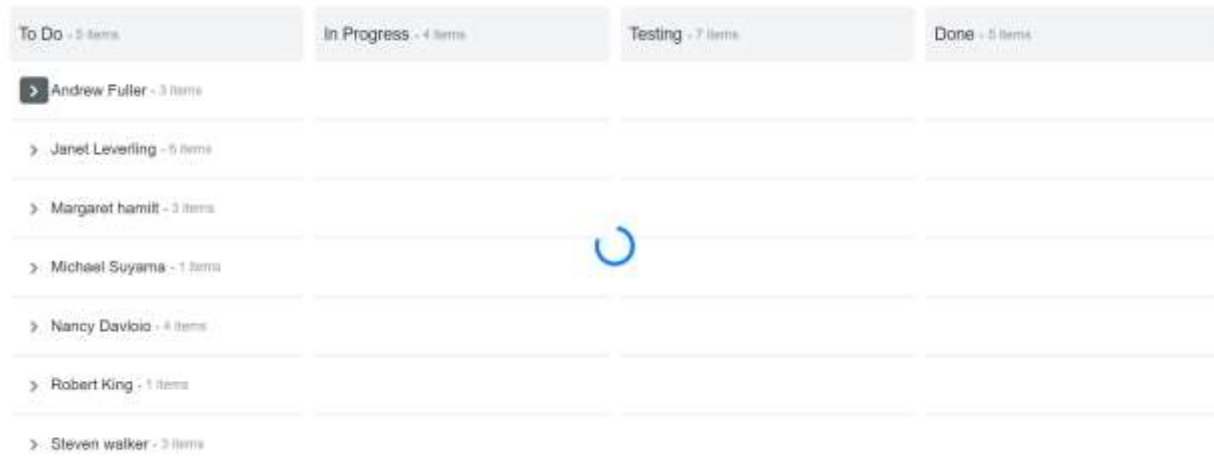
 TaskDetails.Add(new KanbanDataModels { Id = "Task 29", Title
= "Task - 29035", Status = "Testing", Summary = "Fix editing issues
reported in Firefox.", Type = "Bug", Priority = "Critical", Tags =
"Editing,Fix", Estimate = 1.5, Assignee = "Robert King", RankId = 7 });

 TaskDetails.Add(new KanbanDataModels { Id = "Task 30", Title
= "Task - 29036", Status = "Testing", Summary = "Test editing feature in
the IE browser.", Type = "Story", Priority = "Normal", Tags = "Testing",
Estimate = 5.5, Assignee = "Janet Leverling", RankId = 10 });

 return TaskDetails;
 }
}

```

Output will be like the below.



## Split Button

### Getting Started with ASP.NET MVC Split Button Control

This section briefly explains about how to include [ASP.NET MVC Split Button](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
,
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/\_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/\_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Split Button control

Now, add the Syncfusion ASP.NET MVC Split Button control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@model List<object>
@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)Model).Render()
```

#### HOMECONTROLLER.CS

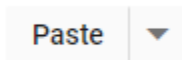
```
public ActionResult Index()
{
 List<object> items = new List<object>();
```

```

items.Add(new
{
 text = "Cut"
});
items.Add(new
{
 text = "Copy"
});
items.Add(new
{
 text = "Paste"
});
return View(items);
}

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Split Button control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

See also

- [SplitButton with icons](#)

## Icons in SplitButton Control

### SplitButton Icons

SplitButton can have an icon to provide the visual representation of the action. To place the icon on a SplitButton, set the [IconCss](#) property to `e-icons` with the required icon CSS. By default, the icon is positioned to the left side of the SplitButton. You can customize the icon's position by using the [IconPosition](#) property.

The following example illustrates how to place icon in SplitButton component.

### CSHTML

```

@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)ViewBag.items).IconCss("e-sb-icons e-paste").Render()
@Html.EJS().SplitButton("element_1").Content("Paste").Items((IEnumerable<object>)ViewBag.items).IconCss("e-sb-icons e-paste").IconPosition(Syncfusion.EJ2.SplitButtons.SplitButtonIconPosition.Top).Render()
<style>
 .e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
 }

 /* csslint ignore:start */
 @@font-face {
 font-family: 'ddb-icons';
 src:

```

```

url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDNtE+dkAAABlAAAAADxnbHlmlh3
3NQAAAdwAAAjMaGVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAADG1heHABEWb1AAABCAAAACBuYW11l1LBM9QAABCgAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDyGPsAAAACAACAAAAAEEAAAFafkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAAPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAAI4AwgEAA5YAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzYmYBHgEXDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHG3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFx06IBkNCQIHCy8bCQG9BwIJDRkgOhoXHwoKGI/+TR1RDyE
OIxo+ExckFAQMFikwVhcMBwYlFRYkBwcMF1YwFCAALDAQUIxcUPhojDiAOUR4cAQvEwWSB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMDg0sOzsaKQ4ONzcnYyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQA+kABQANABCAHwAAARUzFSErAyERiZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGajXUcLP7PDAFNAVl+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41AfbMAZgBJwmYchSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAABAAAAAQAAAAAAQAJAEEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACgAsADCAAQAAAAAACwASAGMAAwABBakAAAACAHU
AAwABBakAAQASAHCAAwABBakAAgAOAIkAAwABBakAAwASAJCAAwABBakABAASAKkAAwABBakABQA
WALsAAwABBakABgASANEAAwABBakACgBYAOMAawABBakACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBJAG8AbgBzAGQAZABiAC0AaQBJAG8AbgBzAFY
AZQByAHMAaQBVAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABLAGQAIAB1AHMAaQBUAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAA
AAAAKAAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxZGZvbG9
OcGFyYSltYXJrLS0tMDMAAA==) format('trueType');

font-weight: normal;
font-style: normal;
}
/* csslint ignore:stop */
.e-sb-icons {
font-family: 'ddb-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
content: '\e701';
}
</style>

```

**ICONBUTTON.CS**

```
using System;
```

```

using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult IconButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
 }
}

```

**Note:** The Essential JS 2 provides a set of icons that can be loaded by applying **e-icons** class name to the element.

You can also use third party icons on the SplitButton using the [IconCss](#) property.

### Vertical Button

Vertical Button in SplitButton can be achieved by adding **e-vertical** class using [CssClass](#) property.

The following example illustrates how to vertical support in SplitButton component.

### CSHTML

```

@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)ViewBag.items).IconCss("e-sb-icons e-paste").CssClass("e-vertical").Render()
<style>
 .e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
 }

 /* csslint ignore:start */
 @@font-face {
 font-family: 'ddb-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAAADxnbH1mlh3
 3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAAYAAAAA

```

```

UbG9jYQGOAegAAAHQAAAADG1heHABEWBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMJntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAAAAADygABAAAAAAAAAAAAAAAAABQABAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAAAAAAAAEAAAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAAI4AwgEAAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBGcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXhGE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHcy8bCQG9BwIJDRkgOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfikwVhcMBwYlFRYkBWcMF1YwFCALDAQUIxcUPhojDiAOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsaKQ4ONZcniyYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQA+kABQANABcAHwAAARUzFSErAYERiZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVl+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEBDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41afbmAZgBJwmYchSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAABAAAAQAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAAABgAJAC4AAQAAAAAACGAsADcAAQAAAAAACWASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmlR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIABlAHMAaQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxZGZvbncQ
OcGFyYSltYXJrLS0tMDMAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.e-sb-icons {
 font-family: 'ddb-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
 content: '\e701';
}
</style>

```

## VERTICALBUTTON.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

```



```

using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult VerticalButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
 }
}

```

## See Also

- [SplitButton popup with icons](#)

## Popup Items

### Icons

The Popup action item have an icon or image to provide visual representation of the action. To place the icon on a popup item, set the [iconCss](#) property to e-icons with the required icon CSS. By default, the icon is positioned to the left side of the popup action item.

In the following sample, the icons for Cut, Copy, Paste, Undo and Redo menu items are added using the iconCss property.

### CSHTML

```

@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)ViewBag.items).IconCss("e-sb-icons e-paste").Render()
<style>
 .e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
 }

 /* csslint ignore:start */
 @@font-face {
 font-family: 'ddb-icons';
 src:
 url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDnE+dkAAABlAAADxnbHlmlh3

```

```
3NQAAAdwAAAJMaGVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAACRobXR4E6AAAAAAYAAAAA
UbG9jYQGOAegAAAHAQAAADG1heHABEWBlAAABCAAAACBuYW1l1LBM9QAABCgAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAABQABAAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAACAACAAAAAEEAAAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAQAptAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAWgPsAJYAAAAABAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAACAAAAAwAAABQAAwABAAAFAAEEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEEAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHCy8bCQG9BwIJDkrgOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfIkWVhCMBwYlFRYkBWcMf1YwFCALDAQUIxcUPhojDiaOUR4cAQvEwwsB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMDg0sOzsaKQ4ONzcniiYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQA+kABQANABcAHwAAARUzFSErAYERizU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVl+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41AfbMAZgBJwmYcHSbA/48A2r8lgNqfgAAAAASAN4AAQAAAAA
AAAAABAAAAQAAAAAAQAJAEEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACGAsADcAAQAAAAAACwASAGMAAwABBAkAAAACAHU
AAwABBAkAAQASAHcAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWljB25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWljB25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2l2b2Zlc2
vAG4AcwBSAGUAZwB1AGwAYQByAGQAZABiAC0AaQBJAG8AbgBzAGQAZABiAC0AaQBJAG8AbgBzAFY
AZQByAHMAaQBVAG4AIAAxAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwB1AG4AZQB
yAGEAdABlAGQAIABlAHMAaQBUAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQB1AHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxZGZvb2N0
OcGFyYS1tYXJrLS0tMDMAAA==) format('trueType');
font-weight: normal;
font-style: normal;
}
/* csslint ignore:stop */
.e-sb-icons {
font-family: 'ddb-icons' !important;
speak: none;
font-size: 55px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.e-paste::before {
content: '\e701';
}
.e-cut::before {
content: '\e700';
}
.e-copy::before {
content: '\e70a';
}
</style>
```

**POPUPICONS.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult PopupIcons()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut",
 iconCss = "e-sb-icons e-cut"
 });
 items.Add(new
 {
 text = "Copy",
 iconCss = "e-icons e-copy"
 });
 items.Add(new
 {
 text = "Paste",
 iconCss = "e-sb-icons e-paste"
 });
 ViewBag.items = items;
 return View();
 }
 }
}

```

## Template

*Item Templating*

Popup items can be customized by using the [beforeItemRender](#) event. The item render event triggers while rendering each Popup action item. The event argument will be used to identify the action item and customize it based on the requirement.

**CSHTML**

```

@Html.EJS().SplitButton("element").Items((IEnumerable<object>)ViewBag.items)
.IconCss("e-sb-icons e-paste").BeforeItemRender("beforeItemRender").Render()
<script>
 function beforeItemRender(args) {
 if (args.item.text === 'Edit') {
 args.element.innerHTML = '<div>Paste
Text<div>Provides option to paste only the
selected
text.</div></div>';
 args.element.style.height = '80px';
 var span = args.element.children[0];
 span.setAttribute('class', 'e-cm-icons e-pastetext e-align');
 }
 }

```

```

 var div = args.element.children[1];
 div.setAttribute('class', 'e-div-align');
 } else {
 args.element.innerHTML = '<div>Paste
Special<div>Provides options to paste formulas,
 values, comments,
validations etc...</div></div>';
 args.element.style.height = '80px';
 var span = args.element.children[0];
 span.setAttribute('class', 'e-cm-icons e-pastespecial e-align');
 var div = args.element.children[1];
 div.setAttribute('class', 'e-div-align');
 }
}
</script>
<style>
 @@font-face {
 font-family: 'e-context-menu';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjJNRVMAAAEoAAAVmNtYXDicOK6AAABjAAAADhnbHlmGcE
PFQAAAcwAAAMwaGVhZA69CA8AAADQAAAANmhoZWEH9AQEAAAArAAAACRobXR4DAAAAAAAAAYAAAA
MbG9jYQDYAZgAAAHEAAAACG1heHABEGDAAAABCAAAACBuYW11xY1dlQAABPwAAAKFcG9zdPJwcMo
AAAEeAAAAASAABAAAEAAAAAFWEAAAAAADlwABAAAAAAAAAAAAAAAAAAAAAAwABAAAAAQAAgmhm8l8
PPPUACwQAAAAAANYD4Y8AAAAA1gPhjwAAAAADlwP0AAAACAACAAAAAAAAAAAAEAAAADALQABQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA4mDiYQQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAAAAAAAgAAAAAAMAAUAMAAQAAABQABAakAAAAABAAEAAEAAOJh//8AAOJg//8AAAAABAAQAAAA
BAAIAAAAAANgBmAFAAAAAAAXA/QABAALAC0ATgCzAAABISchJzcVHwc/By8HDwYBFSE1MxEhESU
HFQ8GLwc/Bx8GJysBDw4RHw4zITM/DhEvDisBLw4rAQ8NAUQBd1xAfr0BAwQGBwgICgkJCacGBAM
BAQMEBgICQkKCAgHBgQD/qYB1l79jQFoAQMEBgCHCQkJCQgGBgQDAQEDBAYGCAkJCQkHBwYEA6y
9CgkICQgHBwCGBQQAawMBAQEBAwMDBQUGBwCHCAkICQoCeAoJCAkIBwCHBgUEBAMDAQEBAQMDAwU
FBgCHBgJCAkKvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQgHBwCfBgQBBYVRuh0FBQkIBwUFAgE
BAgUFBwgJCgkJCacGBAMBAQMEBgICQEI fX39LwLRMwQFCAGHBQUCAQECCBUHCAgJCQkIBwUEAwE
BAwQFBwgJIGICAwQFBQYGBwgICakJCf0pCQkJCAGIBwYGBQUEAwICAgIDBAUFBgYHCAgICQkJAtc
JCQkICAgHBgYFBQQDAgIKCQkICAgHBgYFBAQDAgICAgMEBAUGBgICAgJCQAFAAAAAAXA/QABwA
PABcAOACdAAABHwIjPwIDMzcZfZMDIYcVITUzESERJQCVDwYvBz8HHwYnKwEPDhEfDjMhMz8OES8
OKwEvDisBDw0B/wQKK3MmBQ6dMyeHKDWC090B1l79jQFoAQMEBgCHCQkJCQgGBgQDAQEDBAYGCAk
JCQkHBwYEA6y9CgkICQgHBwCGBQQAawMBAQEBAwMDBQUGBwCHCAkICQoCeAoJCAkIBwCHBgUEBAM
DAQEBAQMDAwUFBgCHBgJCAkKvQqEBgUHBwCICQkJCgoKCwsLCwoKCgkJCQgHBwCfBgQCFREigG4
SM/6wd3cBe/t9ff0vAtEzBAUICAcFBQIBAQIFBQcICakJCQgHBQQDAQEDBAUHCAkiAgIDBAUFBgY
HCAgICQkJ/SkJCQkICAgHBgYFBQQDAgICAgMEBQUGBgICAgJCQkC1wkJCQgICAcGBgUFBAMCAgo
JCQgICAcGBgUEBAMCAgICAwQEBQYGBwgICakJAAAAABIA3gABAAAAAAAAAAAAEAAAABAAAAAABAA8
AAQABAAAAAAACAAcAEAAABAAAAAADAA8AFwABAAAAAAAEAA8AJgABAAAAAAAFAA8ANQABAAAAAA
GAA8AQABAAAAAAAKACwATwABAAAAAALABIAewADAAEEECQAAAAIAjQADAAEEECQABAB4A jwADAAE
ECQACAA4ArQADAAEEECQADAB4AuwADAAEEECQAEAB4A2QADAAEEECQAFABYA9wADAAEEECQAGAB4BDQA
DAAEEECQAKAFgBKwADAAEEECQALACQBgyBDb250ZXh0TWVudSAoMilSZWd1bGFyQ29udGV4dE1lbnU
gKDIpQ29udGV4dE1lbnUgKDIpVmVyc2lubiAxLjBDb250ZXh0TWVudSAoMilGb250IGdlbmVyYXR
lZCBlc2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5jZnVzaW9uLmNvbQAgAEMAbwB
uAHQAZQB4AHQATQB1AG4AdQAgACgAMgApAFIAZQBnAHUAbABhAHIAQwBvAG4AdAB1AHgAdABNAGU
AbgB1ACAkAAyACkAQwBvAG4AdAB1AHgAdABNAGUAbgB1ACAkAAyACkAVgB1AHIAcWBPAG8AbgA
gADEALgAwAEMAbwBuAHQAZQB4AHQATQB1AG4AdQAgACgAMgApAEYAbwBuAHQAIABnAGUAbgB1AHIA
AYQB0AGUAZAAGAHUAcWBPAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIAB
TAHQAdQBkAGkAbwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAA
AAAoAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAMBAgEDAQQAD01UX1Bhc3RlU3B1Y2lhbAxNVF9QYXN
0ZVRleHQAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
 }

```

```

/* csslint ignore:stop */
.e-cm-icons {
 font-family: 'e-context-menu';
 font-style: normal;
 font-variant: normal;
 font-weight: normal;
 line-height: 1;
 text-transform: none;
}
@@font-face {
 font-family: 'sb-icons';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRkAAAEoAAAAVmNtYXDNdE+dkAAABlAAAAADxnbHlmlh3
3NQAAAdwAAAjMaGVhZBKOK9sAAADQAAAAANmhoZWEHeANwAAAArAAAAACRobXR4E6AAAAAAAYAAAA
UbG9jYQGOAegAAAHQAAAADG1heHABEWBlAAABCAAAACBuYW1lL1Bm9QAABCgAAAI9cG9zdMjntbU
AAAZoAAAAUAABAAADUv9qAFoEAAAAAADygABAAAAAAAAAAAAAAAAABQABAAAAQAAojXaQl8
PPPUACwPoAAAAANfSc4gAAAAA19JziAAA//oDygPsAAAAACAACAAAAAAAAAAAAEAAAFkABAAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQPtAZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwNS/2oAwGPsAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAA+gAAAPoAAAAAAACAAAAAwAAABQAAwABAAAAFAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAAI4AwgEAASYAAwAA//oDNQPsAA4AHQBYAAAlHgEOAScmJy4BNz4
BMzIFFgYHBgcGLgE2NzYzMhYBHgEXDgEHDgEHDgIWFxYXFjY3NjQ3PgE3HgEXFhQXHgE3PgE3PgE
uAScuAScuASc+ATc+AQcLASYWAVEfFxo6IBkNCQIHcy8bCQG9BwIJDkkgOhoXHwoKgi/+TR1RDyE
OIxo+ExckFAQMfikwVhcmBwYlFRYkBWCMF1YwFCAALDAQUIxcUPhojDiAOUR4cAQvEwswB6gtDTyc
JCBsSKxYhJ0gWKxIaCQknUEILAYcCf2TPI0w2HBUMdg0sOzsaKQ40NzcniiYXNBgYNBcmiyc3OA8
GHRQaOzssDQ4mFRw2TiLOZGdBA/5vAZEDQQAEEAAAAAQA+KABQANABCAHwAAARUzFSErAYERiZU
jNSEBIREhESMVITUjMyMVITUjNSMC733+iT8B9D4+/oj+igE4AXc//c4++j8BOT+7AbZ8+gF2/ks
Bdz4//ksB9AF2fHw+Pj8AAAIAAAAA7cD6QACACQAAAEhEwMOAQcVITUmJyY1ND8BIRcWFxYVFAc
GKwEVITUmJyYnASMCKP8AguQrOy0BGkIRHREkASstEgEEDhQxEQGaJxUcLP7PDAFNAVL+PHBHCBS
bBgsUKR8wX3owBg4NFgsQGxsDFx1zAyMAAAACAAAAAPKA+oAAgATAAABFxEbDgEHHgEXETMRMxE
zETM1IQL+zPlabpADA5t0f2F+XP41afBMAZgBJwmYchSbA/48A2r8lgNqfgAAAAASAN4AAQAAAA
AAAABAAAAAQAAAAAAQAJAAEAAQAAAAAAAgAHAAoAAQAAAAAAAwAJABEAAQAAAAABAAJABoAAQA
AAAAABQALACMAAQAAAAABgAJAC4AAQAAAAAACgAsADcAAQAAAAAACwASAGMAAwABBAkAAAAACAHU
AAwABBAkAAQASAHCAAwABBAkAAgAOAIkAAwABBAkAAwASAJcAAwABBAkABAASAKkAAwABBAkABQA
WALsAAwABBAkABgASANEAAwABBAkACgBYAOMAawABBAkACwAkATsgZGRiLWlj25zUmVndWxhcmR
kYilpY29uc2RkYilpY29uc1ZlcnNpb24gMS4wZGRiLWlj25zRm9udCBnZW5lcmF0ZWQgdXNpbmc
gU3luY2Zlc2l2b2l2bnZXRybyBTdHVKaW93d3cuc3luY2Zlc2l2b2l2bnZ5b20AIABkAGQAYgAtAGkAYwB
vAG4AcwBSAGUAZwBlAGwAYQByAGQAZABiAC0AaQBjAG8AbgBzAGQAZABiAC0AaQBjAG8AbgBzAFY
AZQByAHMAaQBvAG4AIAAxAAC4AMABkAGQAYgAtAGkAYwBvAG4AcwBGAG8AbgB0ACAAZwBlAG4AZQB
yAGEAdABlAGQAIABlAHMAaQBvAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBIAHQAcgBvACA
AUwB0AHUAZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAgAAAA
AAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgADY3V0CHBhc3RlXzAxZGZvbnQ
OcGFyYSltYXJrLS0tMDMAAA==) format('trueType');
 font-weight: normal;
 font-style: normal;
}
.e-sb-icons {
 font-family: 'sb-icons' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}

```

```

 }
 .e-pastespecial::before {
 content: '\e260';
 }
 .e-pastetext::before {
 content: '\e261';
 }
 .e-paste::before {
 content: '\e701';
 }
 .e-dropdown-popup ul {
 max-width: 400px;
 white-space: nowrap;
 }
 .e-align {
 float: left;
 width: 15%;
 margin-top: 15px;
 font-size: 45px;
 color: grey;
 }
 .e-div-align {
 float: right;
 width: 75%;
 line-height: 23px;
 margin: 0 15px 0 0;
 }
}
</style>

```

### ITEMTEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult ItemTemplate()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Edit"
 });
 items.Add(new
 {
 text = "Cut"
 });
 ViewBag.items = items;
 return View();
 }
 }
}

```

*Popup Templating*

The whole popup can be customized as per the requirement. In the following example, the popup can be customized by handling it in [target](#) property.

**CSHTML**

```
<div id='dropdowntarget' style='width:112px;'>
 <div id= "first">
 <div id='black'></div>
 <div id='red'></div>
 <div id='green'></div>
 <div id='gray'></div>
 <div id='blue'></div>
 <div id='violet'></div>
 <div id='brown'></div>
 <div id='darkgoldenrod'></div>
 <div id='aquamarine'></div>
 </div>
</div>
@Html.EJS().SplitButton("element").Target("#dropdowntarget").IconCss("e-sb-
icons e-color").Render()
<style>
 .shortcut {
 float: right;
 margin-top: 9px;
 padding-left: 30px;
 }
 /* csslint ignore:start */
 @@font-face {
 font-family: 'paint';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRIAAAEoAAAAVmNtYXDNdEOdVAAABiAAAADZnbHlmiZD
+uwAAAcgAAADMaGVhZBKhHQAADQAAAAANmhoZWEHjANrAAAArAAAAACRobXR4B+j/8wAAAYAAAAA
IbG9jYQBMAAAAAHAAAAABmlheHABDgBKAAABCAAAACBuYW1ln6hzswAAApQAAAINcG9zdEkLMmU
AAASkAAAANGABAAADUv9qAFoEAP/z//4D6gABAAAAAAAAAAAAAAAAAAAAgABAAAAAQAAAZfc6F8
PPPUACwPoAAAAANfSn9kAAAAA19Kf2f/z//wD6gPhAAACAACAAAAAAAAAAAAEAAAACAD4AAgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQp0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPhAJYAAAABAAAAAAAAABAAAAAPo//M
AAAACAAAAAwAAABQAAwABAAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAEAAAA
AAAAAZgAAAL/8//8A+oD4QAKAD0AAAEWBgceATclJiQHJTMmNjceARcVJx4BBx4BFQ4BIiYnNDY
3PgEvAS4BIw4BBwEGHgI3AT4BLwE1LgEnDgEDEiRlCgulCxP+8RT+GyYDQFxoZQwTBQEDDxEBJzo
nAREOCQkPJQ4cDBcdAf6oG1a3nx8BWQ4RHKADEg1oWwHTLHVwYVml6Kx1BHEqfwYFqWUHEX4tDAo
cEx0nJx0RHgoVUDQpDgsBFAH+px2guFUaAVkNOiCgCXnhCAWOAAAAAAAAAEgDeAAEAAAAAAAAAAQA
AAEAAAAAAAAAEABQABAAEAAAAAAAAIABwAGAAEAAAAAAAAAMABQANAAEAAAAAAAAABQASAAEAAAAAAAU
ACwAXAAEAAAAAAAYABQAIAAEAAAAAAAOALAAnAAEAAAAAAASAEgBTAAMAAQQJAAAAAgBlaAMAAQQ
JAAEACgBnAAMAAQQJAIAIADgBxAAMAAQQJAAMACgB/AAMAAQQJAAQACgCJAAMAAQQJAAUAFgCTAAM
AAQQJAAAYACgCpAAMAAQQJAAoAWACzAAMAAQQJAAAsAJAELIHBhaW50UmVndWxhcncBhaW50cGFpbmR
WZXJzaW9uIDEuMHBhaW50Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Zlc2lvbiBNZXRYbyBTdHV
kaW93d3cuc3luY2Zlc2lvbi5jb20AIABwAGEAaQBwAHQAUGBlAGcAdQBsaGEAcgBwAGEAaQBwAHQ
AcABhAGkAbgB0AFYAZQByAHMAaQBvAG4AIAAxAAC4AMABwAGEAaQBwAHQAQArgBvAG4AdAAgAGcAZQB
uAGUAcgBhAHQAZQByACAAADQByAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0AZQB0AHI
AbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAgAuAGMAbwBtAAAAAAI
AAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMADHBhaW50LWJlY2tldAAAAA=)
format('trueType');
```

```
 font-weight: normal;
 font-style: normal;
 }
 /* csslint ignore:stop */
 .e-sb-icons {
 font-family: 'paint' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 }
 .e-color::before {
 content: '\e700';
 color: black;
 }

 .e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
 }
 #dropdowntarget {
 border: 0.5px solid grey;
 height: 110px;
 width: 110px;
 }
 #black {
 width: 30px;
 height: 30px;
 background-color: black;
 margin: 5px 5px;
 float: left;
 }
 #red{
 width: 30px;
 height: 30px;
 background-color: red;
 margin: 5px 0px;
 float: left;
 }
 #green{
 width: 30px;
 height: 30px;
 background-color: green;
 margin: 5px 5px;
 float: left;
 }
 #gray{
 width: 30px;
 height: 30px;
 background-color: gray;
 margin: 0px 5px;
 float: left;
 }
}
```



```
#blue{
 width: 30px;
 height: 30px;
 background-color: blue;
 float:left;
}
#violet{
 width: 30px;
 height: 30px;
 background-color: violet;
 margin: 0px 5px;
 float:left;
}

#brown{
 width: 30px;
 height: 30px;
 background-color: brown;
 margin: 5px 5px;
 float:left;
}
#darkgoldenrod{
 width: 30px;
 height: 30px;
 background-color: darkgoldenrod;
 margin: 5px 0px;
 float:left;
}
#aquamarine{
 width: 30px;
 height: 30px;
 background-color: aquamarine;
 margin: 5px 5px;
 float:left;
}
#icon{
 width: 10px;
 height: 10px;
 background-color: aquamarine;
 position: absolute;
}
</style>
```

#### POPUPTEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult PopupTemplate()
 {

```

```

 return View();
 }
}

```

### See Also

- [Popup items grouping](#)

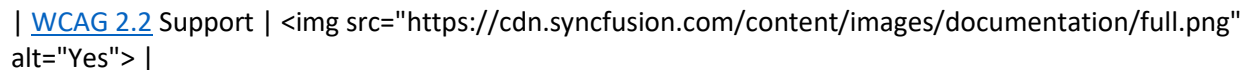
### Accessibility in Split Button Controls

The Split button component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Split button component is outlined below.

| Accessibility Criteria | Compatibility |

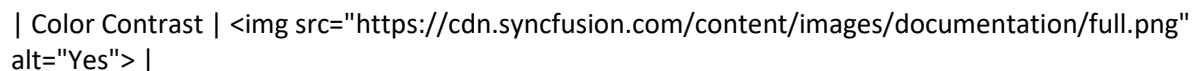
| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" > |

| [Section 508](#) Support |  alt="Yes" > |

| Screen Reader Support |  alt="Yes" > |

| Right-To-Left Support |  alt="Yes" > |

| Color Contrast |  alt="Yes" > |

| Mobile Device Support |  alt="Yes" > |

| Keyboard Navigation Support |  alt="Yes" > |

| [Accessibility Checker](#) Validation |  alt="Yes" > |

| [Axe-core](#) Accessibility Validation |  alt="Yes" > |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The Split button component followed the [WAI-ARIA] patterns to meet the accessibility. The following ARIA attributes are used in the Split button component:

| Attributes | Purpose |

| --- | --- |

| **role** | Indicates the Split button component as **button**, Split button popup as **menu**, and the Split button popup action items as **menuitem**. |

| **aria-haspopup** | Indicates the availability and type of interactive Split button popup element. |

| **aria-expanded** | Indicates whether the Split button popup can be expanded or collapsed, as well as indicates whether its current state is expanded or collapsed. |

| **aria-owns** | Identifies an elements in order to define a visual, functional, or contextual parent/child relationship between DOM elements where the DOM hierarchy cannot be used to represent the relationship. |

| **aria-disabled** | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

### Keyboard interaction

The Split button component followed the [keyboard interaction] guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Split button component.

| **Press** | **To do this** |

| --- | --- |

| **Esc** | Closes the opened popup. |

| **Enter** | Opens the popup, or activates the highlighted item and closes the popup. |

| **Spacer** | Opens the popup. |

| **Up** | Navigates up or to the previous action item. |

| **Down** | Navigates down or to the next action item. |

| **Alt + Up Arrow** | Closes the popup. |

| **Alt + Down Arrow** | Opens the popup. |

### Ensuring accessibility

The Split button component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Split button component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Split button component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

## Styles and Appearances

To modify the SplitButton appearance, you need to override the default CSS of SplitButton component. Find the list of CSS classes and its corresponding section in SplitButton. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

### CSS Class | Purpose of Class

|.e-dropdown-btn|To customize the button part in splitbutton

|.e-split-btn|To customize the dropdown part in split button

|.e-dropdown-popup ul .e-item|To customize the progress button list items

|.e-dropdown-popup ul .e-item .e-menu-icon |To customize the progress button list items icon

## How To

### Create right-to-left SplitButton

SplitButton component has RTL support. This can be achieved by setting [enableRtl](#) as **true**.

The following example illustrates how to enable right-to-left support in SplitButton component.

### CSHTML

```
@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>
t>)ViewBag.items).IconCss("e-sb e-sigma").EnableRtl(true).Render()
<style>
/* csslint ignore:start */
@@font-face {
 font-family: 'sb-icon';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXNlEODVAAABiAAAADZnbHlmFjF
e0gAAAcgAAABIAgVhZBIxLUcAAADQAAAAANmhoZWEHhANTAAAArAAAACRobXR4B+gAAAAAYAAAA
IbG9jYQAKAAAAAHAAAAABmlheHABDQAEAAABCAAAACBuYWllakQFAwAAAhAAAAIlcG9zdEP61+c
AAQ4AAAAAMwABAAADUv9qAFoEAAAAAADbgABAAAAAAAAAAAAAAAAAAAgABAAAAQAAj4iO918
PPPUACwPoAAAAANfSqDwAAAAA19KoPAAAAAADbgPqAAAAACAACAAAAAAAAAAAAEAAAACABIAAQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQF0AZAABQAAANoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
AAAAACAAAAAwAAABQAAwABAAAFAAEACIAAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAEAAAA
AAAAAJAAAAEAAAAAA24D6gARAAATHQETARUhEyMHIQEDIRUzESG65f7gAsolLBH9ywES9gIFLfl
wA5xDQv6u/pBSAQNyAVwBaXIBBAAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAEABwABAAEAAAA
AAAIABwAIAAEAAAAAAAMABwAPAAEAAAAAAQABwAWAAEAAAAAAAUACwAdAAEAAAAAAAYABwAoAAE
AAAAAAoALAAvAAEAAAAAAAsAEgBbAAMAAQQJAAAAAgBtAAMAAQQJAAEADgBvAAMAAQQJAAIADgB
9AAMAAQQJAAAMADgCLAAMAAQQJAAQADgCZAAMAAQQJAAUAFgCnAAMAAQQJAAAYADgC9AAMAAQQJAAo
AWADLAAMAAQQJAAsAJAEjIHNiLWljY25SZWd1bGFyc2ItaWNvbnNiLWljY25WZXJzaW9uIDEuMHN
iLWljY25Gb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAHMAYgAtAGkAYwBvAG4AUgBlAGcAdQBsaGEAcgBzAGIALQBpAGMABwBuAHMA
YgAtAGkAYwBvAG4AVgBlAHIAcWBPAG8AbgAgADEALgAwAHMAYgAtAGkAYwBvAG4ARgBvAG4AdAA
```

```

gAGcAZQBwAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
AZQB0AHIAbwAgAFMAdABlAGQAaQBvAHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMabwB
tAAAAAAIAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMACXN1bW1hdGlvbgAAAA=
=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.e-sb {
 font-family: 'sb-icon' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-sigma::before {
 content: '\e700';
}
.e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
}
</style>

```

## RTL.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult Rtl()
 {
 // define the array of JSON
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Autosum"
 });
 items.Add(new
 {
 text = "Average"
 });
 items.Add(new
 {
 text = "Count numbers"
 });
 items.Add(new

```

```

 {
 text = "Min"
 });
 items.Add(new
 {
 text = "Max"
 });
 ViewBag.items = items;
 return View();
 }
}

```

### Group items in Popup

Grouped items are possible in SplitButton by templating entire popup with ListView. Check [ListView grouping](#) and create such items. Create ListView with id `listview` and provide element of the ListView as target of SplitButton to render it in popup area.

In this following example, ListView is created and its element is set as [target](#) for SplitButton.

### CSHTML

```

@using Syncfusion.EJ2.Lists;
@Html.EJS().ListView("listview").Enable(true).DataSource((IEnumerable<object>)ViewBag.items).Fields(new ListViewFieldSettings { GroupBy = "category" })
.SortOrder(SortOrder.Descending).Render()
@Html.EJS().SplitButton("element").Content("Clipboard").Target("#listview").Render()
<style>
 .e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
 }
</style>

```

### LISTVIEW.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.SplitButtons;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult SplitButton()
 {
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut",
 category = "Basic"
 });
 items.Add(new

```

```

 {
 text = "Copy",
 category = "Basic"
 });
 items.Add(new
 {
 text = "Paste",
 category = "Basic"
 });
 items.Add(new
 {
 text = "Paste as Formula",
 category = "Advanced"
 });
 items.Add(new
 {
 text = "Paste as Hyperlink",
 category = "Advanced"
 });
 ViewBag.items = items;
 return View();
 }
}

```

### Open a dialog on popup item click

This section explains about how to open a dialog on SplitButton popup item click. This can be achieved by handling dialog open in [select](#) event of the SplitButton.

In the following example, Dialog will open while selecting **Update...** item:

### CSHTML

```

@Html.EJS().Dialog("dialog").Header("Software Update").Content("Are you sure
want to
update?").Width("250px").Visible(false).Buttons(ViewBag.button).Render()
@Html.EJS().SplitButton("element").Content("Help").Items((IEnumerable<object
>)ViewBag.items).Select("onSelect").Render()
<script>
 function onSelect(args) {
 if (args.item.text === 'Update...') {
 ej.base.getComponent(document.getElementById('dialog'),
"dialog").show();
 }
 }
 function okBtnClick() {
 ej.base.getComponent(document.getElementById('dialog'),
"dialog").hide();
 }
 function cancelBtnClick() {
 ej.base.getComponent(document.getElementById('dialog'),
"dialog").hide();
 }
</script>
<style>
.e-split-btn-wrapper{

```

```
margin: 20px 20px 5px 5px;
}
</style>
```

### DIALOGBUTTON-CORE.CS

```
public ActionResult DialogButton()
{
 List<object> buttons = new List<object>() { };
 buttons.Add(new
 {
 isPrimary = true,
 cssClass = "e-flat",
 content = "ok",
 click = "okBtnClick"
 });
 buttons.Add(new
 {
 isPrimary = true,
 cssClass = "e-flat",
 content = "cancel",
 click = "cancelBtnClick"
 });
 ViewBag.DialogButtons = buttons;
 return View();
}
```

### Set the disabled state

SplitButton component can be enabled or disabled by disabled property. To disable SplitButton component, set the [disabled](#) property as true.

The following example illustrates how to set the disable state in SplitButton component.

### CSHTML

```
@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)ViewBag.items).IconCss("e-sb e-sigma").Disabled(true).Render()
<style>
/* csslint ignore:start */
@@font-face {
 font-family: 'sb-icon';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRsAAAEoAAAAVmNtYXNlODVAAABiAAAADZnbHlmFjF
eOgAAAcgAAABIAGVhZBIxlUcAAADQAAAAANmhoZWEHhANTAAAArAAAACRobXR4B+gAAAAAYAAAA
IbG9jYQAKAAAAAAHAAAAABmlheHABDQAEAAABCAAAACBuYl1lakQFAwAAAhAAAAIILcG9zdEP61+c
AAQ4AAAAAMwABAAADUv9qAFoEAAAAAADbgABAAAAAAAAAAAAAAAAAAAAAgABAAAAQAAj4iO918
PPPUACwPoAAAAANfSqDwAAAAA19KoPAAAAAADbgPqAAACAACAAAAAAAAAAAAAAEAAAAACABIAAQAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQP0AZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAANS/2oAWgPqAJYAAAABAAAAAAAAABAAAAAPoAAA
AAACAAAAAwAAABQAAwABAAAFAAEACIAAAEAAQAAQAA5wD//wAA5wD//wAAAAEABAAAAEAAAA
AAAAAJAAAAEAAAAAA24D6gARAAATHQETARUhEyMHIQEDIRUzESG65f7gAsolLBH9yWES9gIFLfl
wA5xDQv6u/pBSAQNyAVvBaXIBBAAAAAAAEgDeAAEAAAAAAAAAAQAAAAEAAAAAAEABwABAAEAAAA
AAAIABwAIAEAAAAAAAMABwAPAAEAAAAAAQABwAWAAEAAAAAAAUACwAdAAEAAAAAAAYABwAoAAE
AAAAAAoALAAvAAEAAAAAAASAEgBbAAMAAQQJAaaaaAgBtAAMAAQQJAEEADgBvAAMAAQQJAIIADgB
9AAMAAQQJAAMADgCLAAMAAQQJAQAADgCZAAMAAQQJAUAUFgCnAAMAAQQJAAYADgC9AAMAAQQJAAo
```



```

AWADLAAMAAQQJAAsAJAEjIHNiLWljb25SZWd1bGFyc2ItaWNvbnNiLWljb25WZXJzaW9uIDEuMHN
iLWljb25Gb250IGdlbmVyYXRlZCB1c2luZyBTeW5jZnVzaW9uIE1ldHJvIFN0dWRpb3d3dy5zeW5
jZnVzaW9uLmNvbQAgAHMAYgAtAGkAYwBvAG4AUgBlAGcAdQBsAGEAcgBzAGIALQBpAGMABwBuAHM
AYgAtAGkAYwBvAG4AVgBlAHIAcWBPAG8AbgAgADEALgAwAHMAYgAtAGkAYwBvAG4ARgBvAG4AdAA
gAGcAZQBUAGUAcgBhAHQAZQBkACAAdQBzAGkAbgBnACAAUwB5AG4AYwBmAHUAcwBpAG8AbgAgAE0
AZQB0AHIAbwAgAFMAdAB1AGQAaQBVaHcAdwB3AC4AcwB5AG4AYwBmAHUAcwBpAG8AbgAuAGMABwB
tAAAAAIAAAAAAAACgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAgECAQMACXN1bW1hdGlvbgAAAA=
=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
/* csslint ignore:stop */
.e-sb {
 font-family: 'sb-icon' !important;
 speak: none;
 font-size: 55px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.e-sigma::before {
 content: '\e700';
}
.e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
}
</style>

```

## DISABLED.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace WebApplication1.Controllers
{
 public class SplitButtonController : Controller
 {
 public ActionResult Disabled()
 {
 // Define the array of JSON
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Autosum"
 });
 items.Add(new
 {
 text = "Average"
 });
 items.Add(new

```

```

 {
 text = "Count numbers"
 });
 items.Add(new
 {
 text = "Min"
 });
 items.Add(new
 {
 text = "Max"
 });
 ViewBag.items = items;
 return View();
 }
}

```

### Underline a character in a text

To underline a particular character in a text, it can be handled in [beforeItemRender](#) event by adding `<u>` tag in between the text and given as innerHTML in `li` rendering.

In the following example, **C** is underlined in the text **Copy**:

#### CSHTML

```

@Html.EJS().SplitButton("element").Content("Paste").Items((IEnumerable<object>)ViewBag.items).BeforeItemRender("itemRender").Render()
<script>
 function itemRender(args){
 if (args.item.text === 'Copy') {
 //To underline a particular text.
 args.element.innerHTML = '<u>C</u>opy';
 }
 }
</script>
<style>
 .e-split-btn-wrapper{
 margin: 20px 20px 5px 5px;
 }
</style>

```

#### UNDERLINE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.SplitButtons;
namespace EJ2CoreSampleBrowser.Controllers.Button
{
 public partial class ButtonController : Controller
 {
 public ActionResult SplitButton()
 {

```

```

 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 return View();
 }
}

```

## Migration from Essential JS 1

This article describes the API migration process of SplitButton component from Essential JS 1 to Essential JS 2.

### Properties

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Specifies the text of the splitbutton | **Property:** *text* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login") | **Property:** *content* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("Paste").Render() |

| Popup content | **Property:** *target* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target") | **Property:** *target*

<br/><br/> @Html.EJS().SplitButton("splitbutton").Content("Paste").Target("#target").Render() |

| Popup items | Not applicable | **Property:** *items* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("Paste").Items("ViewBag.items").Render() |

| Arrow position | **Property:** *arrowPosition* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login").Target("#target").ArrowPosition("@ArrowPosition.Left") | Not applicable |

| Button mode | **Property:** *buttonMode* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login").Target("#target").ButtonMode("@ButtonMode.Dropdown") | Not applicable |

| Content type | **Property:** *contentType* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login").Target("#target").ContentType("TextOnly")

| Not applicable |

| Icons | **Property:** *prefixIcon* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login").Target("#target").ContentType("TextAndImage").PrefixIcon("e-icon e-handup") | **Property:** *iconCss* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("Paste").Target("#target").IconCss("e-icons e-paste").Render() |

| Icon position | **Property:** *imagePosition* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login").Target("#target").ContentType("TextAndImage").PrefixIcon("e-icon e-handup").ImagePosition("@ImagePosition.ImageTop") | **Property:** *iconPosition* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("Paste").Item("ViewBag.items").IconCss("e-icons e-paste").IconPosition(Syncfusion.EJ2.SplitButtons.SplitButtonIconPosition.Top).Render() |

| Secondary icon | **Property:** *suffixIcon* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("login").Target("#target").ContentType("TextAndImage").PrefixIcon("e-icon e-handup").SuffixIcon("e-icon e-palette") | Not applicable |

| Adding custom class | **Property:** *cssClass* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").CssClass("custom-class") | **Property:** *cssClass* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").CssClass("custom-class").Render() |

| Disabled state | **Property:** *enabled* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Enabled(false) |

**Property:** *disabled* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Target("#target").Disabled(true).Render() |

| RTL | **Property:** *enableRTL* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").EnableRTL(true) |

**Property:** *enableRtl* <br/><br/>

@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").EnableRtl(true).Render() |

| Height | **Property:** *height* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Height(30) | Not applicable |

| Width | **Property:** *width* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Width(100) | Not applicable |

| HTML attributes | **Property:** *htmlAttributes* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").HtmlAttributes("")

| Not applicable |

| Show rounded corner | **Property:** *showRoundedCorner* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").ShowRoundedCorner(true) | Not applicable |

| Size | **Property:** *size* <br/><br/>

@Html.EJ().SplitButton("splitbutton").Text("Small").Target("#target").Size(ButtonSize.Small) |

**Property:** *cssClass* <br/><br/>

```
@Html.EJS().SplitButton("splitbutton").Content("Small").Item("ViewBag.items").IconCss("e-small").Render() |
```

## Methods

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Destroy method | **Method:** *destroy* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target")
var splitButton = $("#splitbutton").data("ejSplitButton");
splitButton.destroy(); | Method: destroy


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Render()
var splitButton = document.getElementById("splitbutton").ej2_instances[0];
splitButton.destroy(); |
```

| Disable method | **Method:** *disable* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target")
var splitButton = $("#splitbutton").data("ejSplitButton");
splitButton.disable(); | Not applicable |
```

| Enable method | **Method:** *enable* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target")
var splitButton = $("#splitbutton").data("ejSplitButton");
splitButton.enable(); | Not applicable |
```

| Hide popup | **Method:** *hide* <br/><br/>

```
<@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target")
var splitButton = $("#splitbutton").data("ejSplitButton");
splitButton.hide(); | Method: toggle


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Render()
var splitButton = document.getElementById("splitbutton").ej2_instances[0];
splitButton.toggle(); |
```

| Show popup | **Method:** *show* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target")
var splitButton = $("#splitbutton").data("ejSplitButton");
splitButton.show(); | Method: toggle


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Render()
var splitButton = document.getElementById("splitbutton").ej2_instances[0];
splitButton.toggle(); |
```

## Events

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| BeforeOpen event | **Event:** *beforeOpen* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").BeforeOpen("beforeOpen")
function beforeOpen(args) {
 / code block /
} |
```

**Event:** *beforeOpen* <br/><br/>

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").BeforeOpen("beforeOpen").Render()
function beforeOpen(args) {
 / code block /
} |
```

| Click event | **Event:** *click* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Click("click")

function click(args) {
 / code block /
} | Event: click


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Click("click").Re
nder()
function click(args) {
 / code block /
} |
```

| Close event | **Event:** *close* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Close("close")

function close(args) {
 / code block /
} | Event: close


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Close("close")<
br/>function close(args) {
 / code block /
} |
```

| Create event | **Event:** *create* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Create("create")

function create(args) {
 / code block /
} | Event: created


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Created("creat
ed").Render()
function created() {
 / code block /
} |
```

| Destroy event | **Event:** *destroy* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Destroy("destroy")

function destroy(args) {
 / code block */
} | Not applicable
|
```

| ItemMouseOut event | **Event:** *itemMouseOut* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").ItemMouseOut("ite
mMouseOut")
function itemMouseOut(args) {
 / code block */

} | Not applicable |
```

| ItemMouseOver event | **Event:** *itemMouseOver* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").ItemMouseOver("it
emMouseOver")
function itemMouseOver(args) {
 / code block
*/
} | Not applicable |
```

| Item select event | **Event:** *itemSelected* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Item-
selected("itemSelected")
function itemSelected(args) {
 /
code block /
} | Event: select


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Select("select")
.Render()
function select(args) {
 / code block /
} |
```

| Open event | **Event:** *open* <br/><br/>

```
@Html.EJ().SplitButton("splitbutton").Text("SplitButton").Target("#target").Open("open")

function open(args) {
 / code block /
} | Event: open


```

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").Open("open").
Render()
function open(args) {
 / code block /
} |
```

|                   |                |                                  |
|-------------------|----------------|----------------------------------|
| BeforeClose event | Not applicable | <b>Event:</b> <i>beforeClose</i> |
|-------------------|----------------|----------------------------------|

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").Item("ViewBag.items").BeforeClose("beforeClose").Render()
function beforeClose(args) {
 /
code block */
}
```

|                        |                |                                                  |
|------------------------|----------------|--------------------------------------------------|
| BeforeItemRender event | Not applicable | <b>Event:</b> <i>beforeItemRender</i> <br/><br/> |
|------------------------|----------------|--------------------------------------------------|

```
@Html.EJS().SplitButton("splitbutton").Content("SplitButton").BeforeItemRender("beforeItemRender").Item("ViewBag.items").Render()
function beforeItemRender(args) {

 // ...
} / code block */
}
```

## Splitter

## Getting Started with ASP.NET MVC Splitter Control

This section briefly explains about how to include [ASP.NET MVC Splitter](#) control in your ASP.NET MVC application using Visual Studio.

## Prerequisites

## System requirements for ASP.NET MVC controls

## Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

## Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

## PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

## Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

•

&lt;namespaces&gt;

```
<add namespace="Syncfusion.EJ2"/>
```

&lt;/namespaces&gt;

•

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ \_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Splitter control

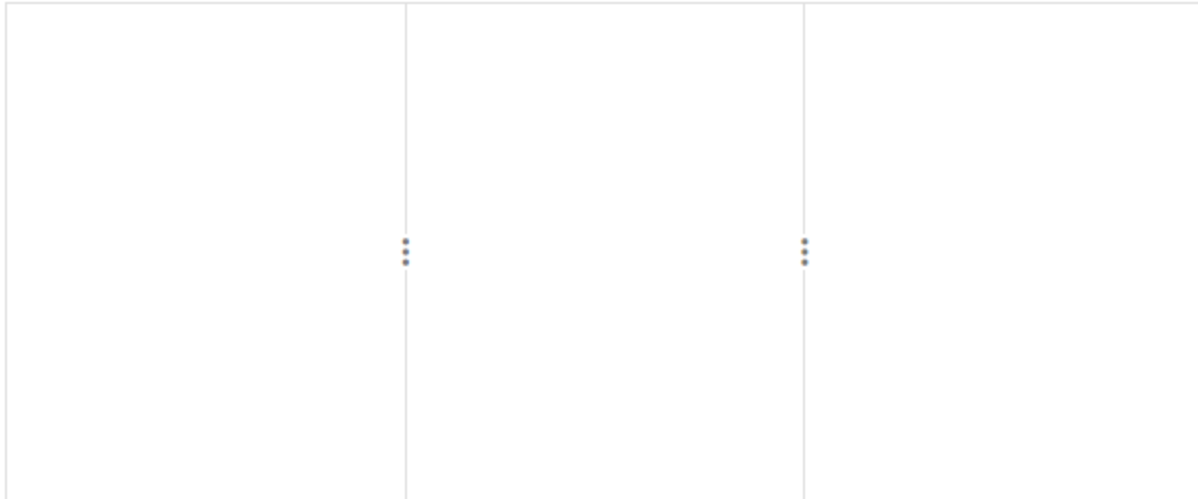
Now, add the Syncfusion ASP.NET MVC Splitter control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("250px").PaneSettings
(item => {
 item.Content("<div></div>").Add(); item.Content("<div></div>").Add(); item.Con
tent("<div></div>").Add(); }).Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Splitter control will be rendered in the default web browser.





### Load content to the pane

You can load the pane contents in either HTML element or string types using [Content](#) property.

#### CSHTML

```
@Html.EJS().Splitter("splitter").Width("550px").Height("250px").PaneSettings
(item => {
 item.Content("<div class='content'><h3 class='h3'>HTML</h3><div
class='code-preview'>#60;!DOCTYPE
html#62;</div>#60;html#62;</div><div>#60;body#6
2;</div>#60;div id='custom-image'><div
style='margin-left: 5px'>#60;img
src='src/albert.png'></div><div>#60;div#62;</div><div>#6
0;/body#62;</div><div>#60;/html#62;</div></div>
</div>").Add();
 item.Content("<div class='content'><h3 class='h3'>CSS</h3><div
class='code-preview'>img {<div id='code-text'>margin:0
auto;</div><div id='code-text'>display:flex;</div><div
id='code-text'>height:70px;</div>
}</div></div>").Add();
 item.Content("<div class='content'><h3 class='h3'>JavaScript</h3><div
class='code-preview'>var image =
document.getElementById('custom-image');<div>image.addEventListener('click',
function() {</div><div style='padding-left: 20px;'>// Code block for click
action</div> }</div></div>").Add();}).Render()
<style>
#code-text {
 margin-left: 5px;
}
.code-preview {
 margin-top: 15px;
 font-size: 12px;
}
.h3 {
 font-size: 14px;
 margin: 4px;
}
.content {
 padding: 12px;
```

```

 }
 .splitter-image {
 margin: 0 auto;
 display: flex;
 height: 115px;
 margin-top: 10px;
 }
</style>

```

| HTML                                                                                                                                                                                                       | CSS                                                                  | JavaScript                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> &lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;div id="custom-image"&gt;   &lt;img src="https://ej2.syncfusion.com   /demos/src/albert.png"&gt; &lt;/div&gt; &lt;/body&gt; &lt;/html&gt; </pre> | <pre> img {   margin:0 auto;   display:flex;   height:70px; } </pre> | <pre> var image = document.getElementById("custom- image"); image.addEventListener("click", function() {   // Code block for click action } </pre> |

### Configure pane size

You can specify the size to each pane using [Size](#) property. It accepts pane values in percentage and pixel types.

In case of pane size is not declared, panes will equally share the sizes automatically.

**Note:** For Horizontal orientation, panes will share the total width.

<br/> For Vertical orientation, panes will share the total height.

### CSHTML

```

@Html.EJS().Splitter("splitter").Width("550px").Height("250px").PaneSettings
(item => {
 item.Size("40%").Content("<div class='content'><h3 class='h3'>HTML</h3><div
 class='code-preview'><!DOCTYPE
 html></div><html></div><div><body&#
 62;</div><div id='custom-image'><div
 style='margin-left: 5px'><img
 src='src/albert.png'></div><div><div></div><div>
 0;/body></div><div></html></div></div>
 </div>").Add();
 item.Size("20%").Content("<div class='content'><h3 class='h3'>CSS</h3><div
 class='code-preview'>img {<div id='code-text'>margin:0
 auto;</div><div id='code-text'>display:flex;</div><div
 id='code-text'>height:70px;</div>
 }</div></div>").Add();
 item.Size("40%").Content("<div class='content'><h3
 class='h3'>JavaScript</h3><div class='code-preview'>var image =
 document.getElementById('custom-image');<div>image.addEventListener('click',

```

```
function() {</div><div style='padding-left: 20px;'> // Code block for click
action</div> }</div></div>").Add();}).Render()
<style>
 #code-text {
 margin-left: 5px;
 }
 .code-preview {
 margin-top: 15px;
 font-size: 12px;
 }
 .h3 {
 font-size: 14px;
 margin: 4px;
 }
 .content {
 padding: 12px;
 }
 .splitter-image {
 margin: 0 auto;
 display: flex;
 height: 115px;
 margin-top: 10px;
 }
</style>
```

### Resizable panes

Splitter allows you to change the pane dimensions by resizing the panes. By default, all the panes are resizable. You can disable the resize by using [Resizable](#) property in each pane settings.

### CSHTML

```
@Html.EJS().Splitter("splitter").Width("550px").Height("250px").PaneSettings
(item => {
 item.Size("40%").Min("30px").Resizable(false).Content("<div
class='content'><h3 class='h3'>HTML</h3><div class='code-
preview'><!DOCTYPE
html></div><html></div><div><body&#
62;</div><div id='custom-image'><div
style='margin-left: 5px'><img
src='src/albert.png'></div><div><div></div><div>
0;/body></div><div></html></div></div>
</div>").Add();
 item.Size("20%").Min("15%").Content("<div class='content'><h3
class='h3'>CSS</h3><div class='code-preview'>img {<div
id='code-text'>margin:0 auto;</div><div id='code-
text'>display:flex;</div><div id='code-
text'>height:70px;</div> }</div></div>").Add();
 item.Size("40%").Content("<div class='content'><h3
class='h3'>JavaScript</h3><div class='code-preview'>var image =
document.getElementById('custom-image');<div>image.addEventListener('click',
function() {</div><div style='padding-left: 20px;'> // Code block for click
action</div> }</div></div>").Add();}).Render()
```

| HTML                                                                                                                                                                                                     | CSS                                                                | JavaScript                                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;div id="custom-image"&gt;   &lt;img src="https://ej2.syncfusion.com   /demos/src/albert.png"&gt; &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre> | <pre>img {   margin:0 auto;   display:flex;   height:70px; }</pre> | <pre>var image = document.getElementById("custom- image"); image.addEventListener("click", function() {   // Code block for click action }</pre> |

### Set minimum and maximum pane sizes

Splitter allows you to set the minimum and maximum sizes for each pane. Resize will happens up to the minimum and maximum values only. It will accepts the size in both percentage and pixel formats.

### CSHTML

```
@Html.EJS().Splitter("splitter").Width("550px").Height("250px").PaneSettings
(item => {
 item.Size("40%").Min("30px").Max("60%").Resizable(false).Content("<div
 class='content'><h3 class='h3'>HTML</h3><div class='code-
 preview'><!DOCTYPE
 html><div><html></div><div><body&#
 62;</div><div id='custom-image'><div
 style='margin-left: 5px'><img
 src='src/albert.png'></div><div><div></div><div>
 0;/body></div><div></html></div></div>
 </div>").Add();
 item.Size("20%").Min("15%").Max("40%").Content("<div class='content'><h3
 class='h3'>CSS</h3><div class='code-preview'>img {<div
 id='code-text'>margin:0 auto;</div><div id='code-
 text'>display:flex;</div><div id='code-
 text'>height:70px;</div> }</div></div>").Add();
 item.Size("40%").Content("<div class='content'><h3
 class='h3'>JavaScript</h3><div class='code-preview'>var image =
 document.getElementById('custom-image');<div>image.addEventListener('click',
 function() {</div><div style='padding-left: 20px;'>// Code block for click
 action</div> }</div></div>").Add();}).Render();
```

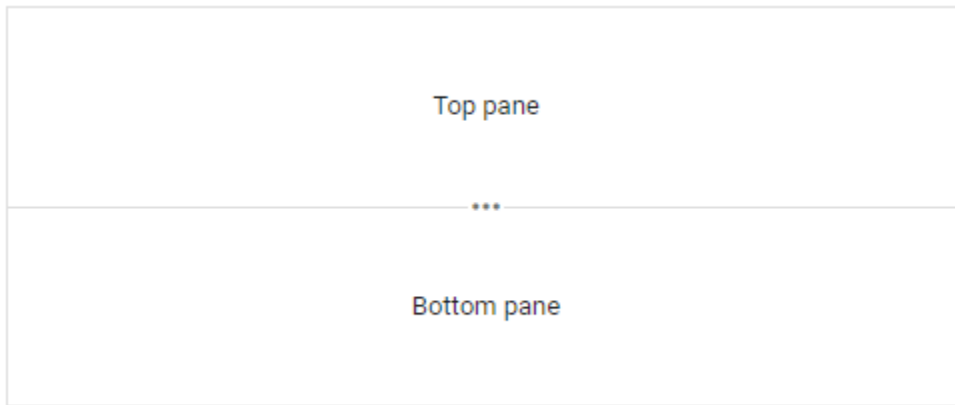
### Orientation

Splitter supports both **Horizontal** and **Vertical** orientation for the panes. By default, it will be rendered in **Horizontal** orientation. You can change it by using [Orientation](#) property.

### CSHTML

```
@using Syncfusion.EJ2.Layouts
@Html.EJS().Splitter("vertical").CssClass("splitterContent").Width("100%").H
eight("200px").PaneSettings(item => {
 item.Size("50%").Min("30%").Content("<div class='content'> Top
 pane</div>").Add();
```

```
item.Size("50%").Min("30%").Content("<div class='content'> Bottom
pane</div>").Add(); }).Orientation(Orientation.Vertical).Render()
<style>
 .content {
 justify-content: center;
 text-align: center;
 align-items: center;
 height: 100%;
 display: flex;
 }
</style>
```



### Nested Splitter

Splitter provides support to render the nested pane to achieve the complex layouts. You can render the nested splitter using the `<div>` element provided for parent splitter.

Also you can render the nested splitter using direct child of the splitter pane. For this, nested splitter should have `100%` width and height to match with the parent pane dimensions.

### CSHTML

```
@using Syncfusion.EJ2.Layouts
@{
 var pane4Content = @"<div class='content'>
 <h3 class='h3'>Preview of sample</h3>
 <div class='splitter-image'>
 <img class='img1'
src='https://ej2.syncfusion.com/demos/src/listview/images/albert.png'
style='width: 20%;margin: 0 auto;' />
 </div>
 </div>";
}
<div>

@Html.EJS().Splitter("outerSplitter").Created("onCreate").PaneSettings(item
=> {
 item.Size("50%").Min("30%").Add();
 item.Size("50%").Content(@pane4Content).Min("30%").Add();
}).Height("430px").Width("600px").Orientation(Orientation.Vertical).Render()
</div>
<style>
```

```

.h3 {
 font-size: 14px;
 margin: 4px;
}
.content {
 padding: 12px;
}
.splitter-image {
 margin: 0 auto;
 display: flex;
 height: 115px;
 margin-top: 10px;
}
#code-text {
 margin-left: 5px;
}
.code-preview {
 margin-top: 15px;
 font-size: 12px;
}
</style>
<script>
 var panelContent = "<div><div class='content'><h3
class='h3'>HTML</h3><div class='code-preview'><!DOCTYPE
html><div><html></div>" +
 "<div><body></div><div id='custom-
image'><div style='margin-left: 5px'><img
src='src/albert.png'></div>" +

"<div><div></div><div></body></div><di
v></html></div></div></div></div>";
 var pane2Content = "<div><div class='content'><h3
class='h3'>CSS</h3><div class='code-preview'>img {<div
id='code-text'>margin:0 auto;</div>" +
 "<div id='code-text'>display:flex;</div><div id='code-
text'>height:70px;</div></div></div>";
 var pane3Content = "<div><div class='content'><h3
class='h3'>JavaScript</h3><div class='code-preview'>var " +
 "image = document.getElementById('custom-
image');<div>image.addEventListener('click', function() {</div>" +
 "<div style='padding-left: 20px;'>// Code block for click
action</div> }</div></div></div>";
 function onCreate(args) {
 document.getElementById('outerSplitter').querySelector('.e-pane-
vertical').setAttribute('id', 'Innersplitter');
 var splitterObj = new ej.layouts.Splitter({
 height: '250px',
 paneSettings: [
 { size: '40%', min: '23%', resizable: false, content:
panelContent },
 { size: '20%', min: '15%', content: pane2Content },
 { size: '40%', content: pane3Content }
],
 width: '100%'
 });
 splitterObj.appendTo('#Innersplitter');
 }

```

```
</script>
```

| HTML                                                                                                                                                                                                     | CSS                                                                | JavaScript                                                                                                                                        |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>&lt;!DOCTYPE html&gt; &lt;html&gt; &lt;body&gt; &lt;div id="custom-image"&gt;   &lt;img src="https://ej2.syncfusion.com   /demos/src/albert.png"&gt; &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre> | <pre>img {   margin:0 auto;   display:flex;   height:70px; }</pre> | <pre>var image = document.getElementById("custom- image"); image.addEventListener("click", function() {   // Code block for click action })</pre> |
| ...                                                                                                                                                                                                      |                                                                    |                                                                                                                                                   |
| <p><b>Preview of sample</b></p>                                                                                        |                                                                    |                                                                                                                                                   |

**Note:** [View Sample in GitHub.](#)

See also

- [Real time example using Splitter.](#)
- [Construct different layouts using Splitter.](#)

## Split panes

This section explain about split panes behaviours.

### Horizontal layout

By default, splitter will render in horizontal orientation. Splitter container will be splitted as panes in horizontal flow direction with vertical seperator.

### CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("250px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'><h4
class='h4'>Grid</h4>The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Schedule
</h4>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
```

```
calendar features, thus allowing users to manage their time
efficiently.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Chart
</h4>ASP.NET charts, a well-crafted easy-to-use charting package, is used to
add beautiful charts in web and mobile applications").Add();
}).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 }
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.

| Grid                                                                                                           | Schedule                                                                                                                                      | Chart                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format. | The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently. | ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications |

### Vertical layout

By setting [Orientation](#) API as `Vertical`, splitter will render in vertical orientation. Splitter container will be splitted as panes in vertical flow direction with horizontal separator.

### CSHTML

```
@using Syncfusion.EJ2.Layouts
@Html.EJS().Splitter("splitter").Width("500px").Height("300px").PaneSettings
(item => {
```



```

 item.Size("100px").Content("<div class='content'><h4
class='h4'>Grid</h4>The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.</div>").Add();
 item.Size("100px").Content("<div class='content'><h4 class='h4'>Schedule
</h4>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
calendar features, thus allowing users to manage their time
efficiently.</div>").Add();
 item.Size("100px").Content("<div class='content'><h4 class='h4'>Chart
</h4>ASP.NET charts, a well-crafted easy-to-use charting package, is used to
add beautiful charts in web and mobile applications").Add();
 }).Orientation(Orientation.Vertical).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 }
</style>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.

## Grid

The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.

## Schedule

The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently.

## Chart

ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications

### Multiple panes

You can render the multiple panes with both **Horizontal/Vertical** orientations.

**CSHTML**

```
@Html.EJS().Splitter("splitter").Width("600px").Height("280px").PaneSettings
(item => {
 item.Size("150px").Content("<div class='content'><h4 class='h4'>PARIS
</h4>Paris, the city of lights and love - this short guide is full of ideas
for how to make the most of the romanticism...</div>").Add();
 item.Size("150px").Content("<div class='content'><h4 class='h4'>CAMEMBERT
</h4>The village in the Orne département of Normandy where the famous French
cheese is originated from.</div>").Add();
 item.Size("150px").Content("<div class='content'><h4 class='h4'>GRENOBLE
</h4>The capital city of the French Alps and a major scientific center
surrounded by many ski resorts, host of the Winter Olympics in
1968.</div>").Add();
 item.Size("150px").Content("<div class='content'><h4 class='h4'>AUSTRALIA
</h4>Australia is a country and continent surrounded by the Indian and
Pacific oceans. Its major cities - Sydney, Brisbane, Melbourne, Perth,
Adelaide - are coastal. Its capital, Canberra, is inland.</div>").Add();
}).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 font-size: 14px;
 }
</style>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.

| PARIS                                                                                                                 | CAMEMBERT                                                                                          | GRENOBLE                                                                                                                               | AUSTRALIA                                                                                                                                                                                          |
|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Paris, the city of lights and love - this short guide is full of ideas for how to make the most of the romanticism... | The village in the Orne département of Normandy where the famous French cheese is originated from. | The capital city of the French Alps and a major scientific center surrounded by many ski resorts, host of the Winter Olympics in 1968. | Australia is a country and continent surrounded by the Indian and Pacific oceans. Its major cities – Sydney, Brisbane, Melbourne, Perth, Adelaide – are coastal. Its capital, Canberra, is inland. |

## Separator

By default, pane separator will be render with **1px** width/height. You can customize the separator size by using [SeparatorSize](#) API.

**Note:** For Horizontal orientation, it will be considered as separator width.

<br/> For Vertical orientation, it will be considered as separator height.

## CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("250px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'><h4
class='h4'>Grid</h4>The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Schedule
</h4>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
calendar features, thus allowing users to manage their time
efficiently.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Chart
</h4>ASP.NET charts, a well-crafted easy-to-use charting package, is used to
add beautiful charts in web and mobile applications").Add();
}).SeparatorSize(5).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 font-size: 15px;
 }
</style>
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
```

```
public ActionResult Index()
{
 return View();
}
```

Output be like the below.

|                                                                                                                                          |                                                                                                                                                                             |                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Grid</b></p> <p>The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.</p> | <p><b>Schedule</b></p> <p>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently.</p> | <p><b>Chart</b></p> <p>ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications</p> |
|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

### Nested Splitter

Splitter provides support to render the nested pane to achieve the complex layouts. You can use the same `<div>` element for splitter pane and nested splitter.

**Note:** Also you can render the nested splitter using direct child of the splitter pane. For this, nested splitter should have **100%** width and height to match with the parent pane dimensions.

### CSHTML

```
<div>

@Html.EJS().Splitter("Horizontal_splitter").Created("onCreate").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'><h4
class='h4'>PARIS </h4>Paris, the city of lights and love - this short guide
is full of ideas for how to make the most of the
romanticism...</div>").Add();
 item.Size("200px").Content("<div class='content'><h4
class='h4'>CAMEMBERT </h4>The village in the Orne département of Normandy
where the famous French cheese is originated from.</div>").Add();
 item.Size("200px").Content("<div id ='vertical_splitter'
class='vertical_splitter'><div class='content'><h4 class='h4'>GRENOBLE
</h4>The capital city of the French Alps and a major scientific center
surrounded by many ski resorts, host of the Winter Olympics in
1968.</div><div class='content'><h4 class='h4'>AUSTRALIA </h4>Australia is a
country and continent surrounded by the Indian and Pacific oceans. Its major
cities - Sydney, Brisbane, Melbourne, Perth, Adelaide - are coastal. Its
capital, Canberra, is inland.</div></div>").Add();
}).Height("320px").Width("600px").Render()
</div>
```

```
<style>
 .content {
 padding: 9px;
 }
 .vertical_splitter.e-splitter.e-splitter-vertical {
 border: none;
 }
 .h4 {
 font-weight: 550;
 font-size: 14px;
 }
</style>
<script type="text/javascript">
 function onCreate(args) {
 var splitterObj = new ej.layouts.Splitter({
 paneSettings: [
 { size: '150px', min: '20%' },
 { size: '100px', min: '20%' }
],
 orientation: 'Vertical'
 });
 splitterObj.appendTo('#vertical_splitter');
 }
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.

|                                                                                                                                       |                                                                                                                        |                                                                                                                                                                                                                        |
|---------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PARIS</b><br>Paris, the city of lights and love - this short guide is full of ideas for how to make the most of the romanticism... | <b>CAMEMBERT</b><br>The village in the Orne département of Normandy where the famous French cheese is originated from. | <b>GRENOBLE</b><br>The capital city of the French Alps and a major scientific center surrounded by many ski resorts, host of the Winter Olympics in 1968.                                                              |
|                                                                                                                                       | ⋮                                                                                                                      | <b>AUSTRALIA</b><br>Australia is a country and continent surrounded by the Indian and Pacific oceans. Its major cities – Sydney, Brisbane, Melbourne, Perth, Adelaide – are coastal. Its capital, Canberra, is inland. |

#### Add or remove pane

You can add the panes programmatic but it will makes you complex. For this, you can use the `addPane/removePane` methods to add and remove the panes dynamically in the splitter.

#### Add pane

You can add the panes dynamically in the splitter by passing [PaneSettings](#) along with index to the `addPane` method

#### CSHTML

```
<div>

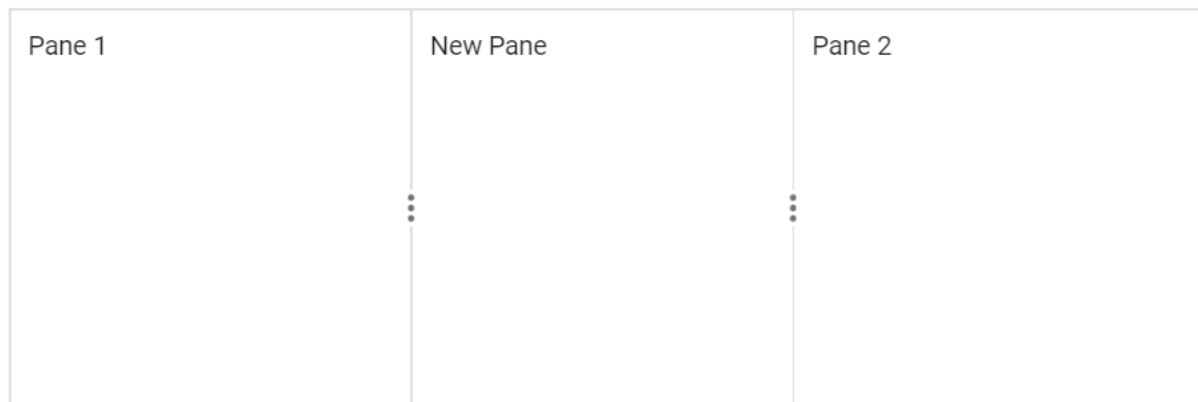
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("190px").Content("<div class='content'>Pane
1</div>").Add();
 item.Size("190px").Content("<div class='content'>Pane
2</div>").Add();
}).Render()
 <button id="addPane" class="e-btn" onclick="onClicked()">Add
pane</button>
</div>
<style>
 .content {
 padding: 9px;
 }
 #addPane {
 margin-top: 15px;
 }
</style>
<script type="text/javascript">
 function onClicked() {
 var splitterObj = document.getElementById('splitter');
```

```
var paneDetails = {
 size: '190px',
 content: '<div class="content">New Pane</div>',
 min: '30px',
 max: '250px',
}
splitterObj.ej2_instances[0].addPane(paneDetails, 1);
}
</script>
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.



ADD PANE

[Remove pane](#)

You can remove the split panes dynamically by passing the pane index to `removePane` method.

## CSHTML

```
<div>

@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'>Pane
1</div>").Add();
 item.Size("200px").Content("<div class='content'>Pane
2</div>").Add();
 item.Size("200px").Content("<div class='content'>Pane
3</div>").Add();
}).Render()
```

```

<button id="removePane" class="e-btn" onclick="onClicked()">Remove
pane</button>
</div>
<style>
 .content {
 padding: 9px;
 }
 #removePane {
 margin-top: 15px;
 }
</style>
<script type="text/javascript">
 function onClicked() {
 var splitterObj = document.getElementById('splitter');
 if
(!ej.base.isNullOrUndefined(document.querySelector('#splitter').querySelecto
rAll('.e-pane-horizontal')[1])) {
 splitterObj.ej2_instances[0].removePane(1);
 }
 }
</script>

```

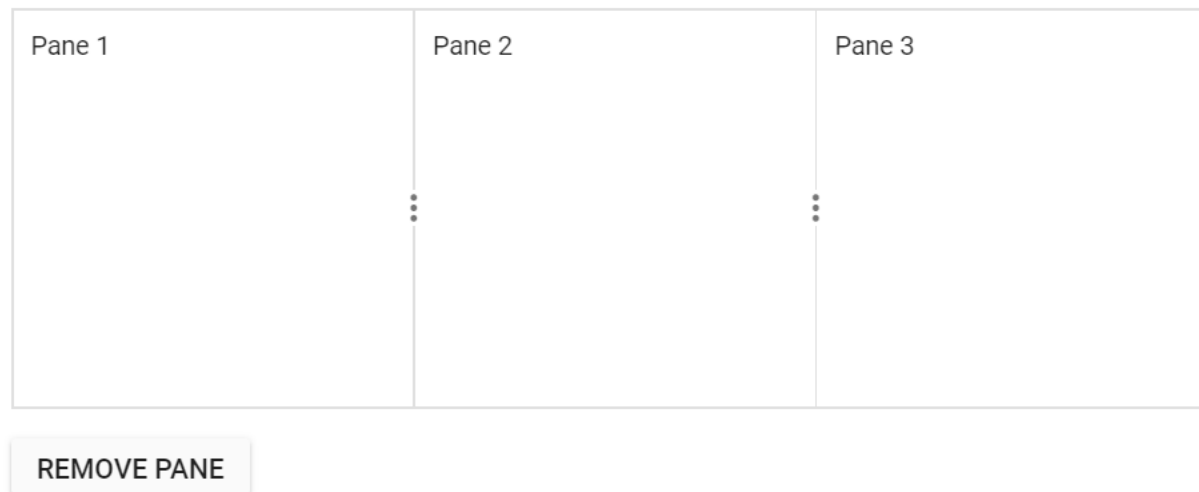
### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.



See Also

- [Resizable split panes](#)



- [Collapsible panes](#)
- [Define size to a panes](#)
- [Specify content to a panes](#)

## Expand and Collapse

### Collapsible panes

The Splitter panes can be configured with built-in expand and collapse functionalities. By default, the **Collapsible** behavior is disabled. Enable the **Collapsible** behavior in the [PaneSettings](#) property to show or hide the expand or collapse icons in the panes. You can dynamically expand and collapse the panes by the corresponding icons.

The following code shows how to enable collapsible behavior.

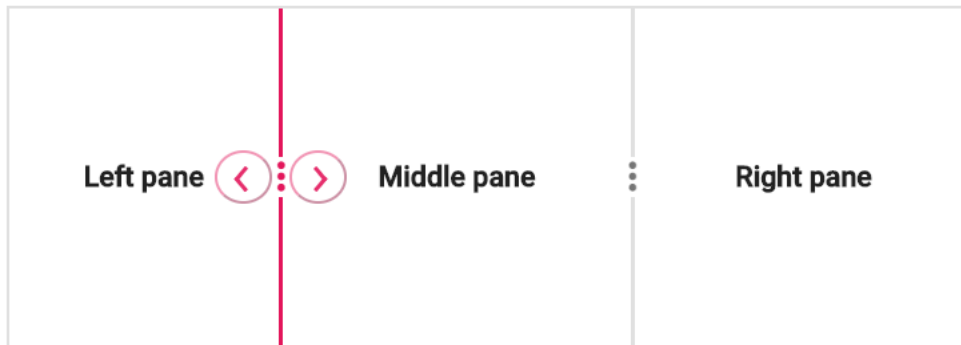
### CSHTML

```
@Html.EJS().Splitter("splitter").Width("500px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'>Left
pane</div>").Collapsible(true).Add();
 item.Size("200px").Content("<div class='content'>Middle
pane</div>").Collapsible(true).Add();
 item.Size("200px").Content("<div class='content'>Right
pane</div>").Collapsible(true).Add();
}).Render()
<style>
 .content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: grid;
 height: 100%;
 }
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Execution of above code's output will be as given below,



### Programmatically control the expand and collapse action

The Splitter provides public method to control the expand and collapse behavior programmatically using the `expand` and `collapse` methods. Refer to the following code sample.

#### CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'>Left
pane</div>").Collapsible(true).Add();
 item.Size("200px").Content("<div class='content'>Middle
pane</div>").Collapsible(true).Add();
 item.Size("200px").Content("<div class='content'>Right
pane</div>").Collapsible(true).Add();
}).Created("onCreated").Render()
@Html.EJS().Button("expand").Content("Expand").Render()
@Html.EJS().Button("collapse").Content("Collapse").Render()
<script>
 var splitterObj;
 function onCreated() {
 splitterObj = this;
 }
 function expandAction() {
 splitterObj.expand(0);
 }
 function collapseAction() {
 splitterObj.collapse(0);
 }
</script>
<style>
 .content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: grid;
 height: 100%;
 }
</style>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Specify initial state to panes

You can render specific panes with collapsed state on page load. Specify a Boolean value to the [collapsed](#) property to control this behavior. The following code explains how to collapse panes on rendering (the second panes renders with collapsed state).

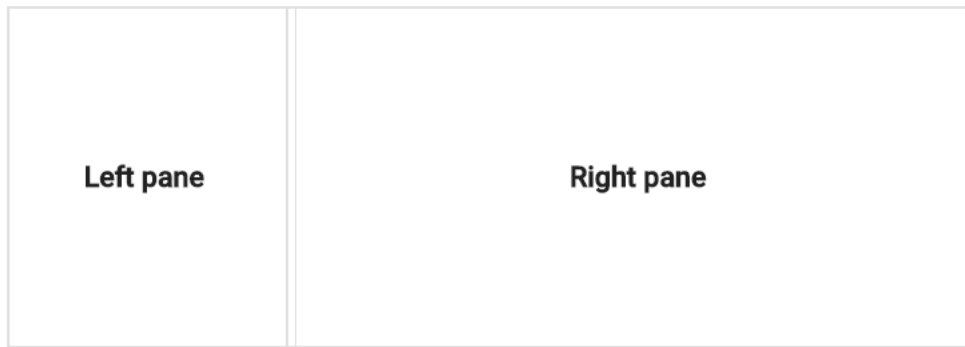
### CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'>Left
pane</div>").Collapsible(true).Add();
 item.Size("200px").Content("<div class='content'>Middle
pane</div>").Collapsed(true).Add();
 item.Size("200px").Content("<div class='content'>Right
pane</div>").Collapsible(true).Add();
}).Render()
<style>
 .content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: grid;
 height: 100%;
 }
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Execution of above code's output will be as given below,



See Also

- [Resizable split panes](#)

## Resize

By default, resizing will be enable for split panes. Resizing gripper element will be add to the separator to makes the resize easy.

**Note:** Horizontal splitter will allows to resize in horizontal directions.

<br/> Vertical splitter will allows to resize in vertical directions.

While resizing, previous and next panes will be adjust its dimensions automatically.

## Min and Max validation

Splitter allows you to set the minimum and maximum sizes for each pane. Resizing will not be occur over the minimum and maximum values.

## CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Min("20%").Max("40%").Content("<div
class='content'><h4 class='h4'>Grid</h4>The ASP.NET DataGrid control, or
DataTable is a feature-rich control used to display data in a tabular
format.</div>").Add();
 item.Size("200px").Min("30%").Max("60%").Content("<div
class='content'><h4 class='h4'>Schedule </h4>The ASP.NET Scheduler, a.k.a.
event calendar, facilitates almost all calendar features, thus allowing
users to manage their time efficiently.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Chart
</h4>ASP.NET charts, a well-crafted easy-to-use charting package, is used to
add beautiful charts in web and mobile applications").Add();
}).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 }
</style>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

## Prevent resizing

You can disable the resizing for the pane by setting `false` to the [Resizable](#) API within [PaneSettings](#).

**CSHTML**

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Min("20%").Max("40%").Resizable(false).Content("<div
class='content'><h4 class='h4'>Grid</h4>The ASP.NET DataGrid control, or
DataTable is a feature-rich control used to display data in a tabular
format.</div>").Add();
 item.Size("200px").Min("30%").Max("60%").Content("<div
class='content'><h4 class='h4'>Schedule </h4>The ASP.NET Scheduler, a.k.a.
event calendar, facilitates almost all calendar features, thus allowing
users to manage their time efficiently.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Chart
</h4>ASP.NET charts, a well-crafted easy-to-use charting package, is used to
add beautiful charts in web and mobile applications").Add();
}).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 }
</style>
```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.

|                                                                                                                                          |                                                                                                                                                                             |                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p><b>Grid</b></p> <p>The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format.</p> | <p><b>Schedule</b></p> <p>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently.</p> | <p><b>Chart</b></p> <p>ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications</p> |
|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|

### Refresh content on resizing

While resizing the panes, you can refresh the pane contents by using either [ResizeStart](#), [Resizing](#) or [ResizeStop](#) events.

### Customize the resize grip and cursor

You can customize the resize gripper icon and cursor in css level.

### CSHTML

```
<div class="control_wrapper">

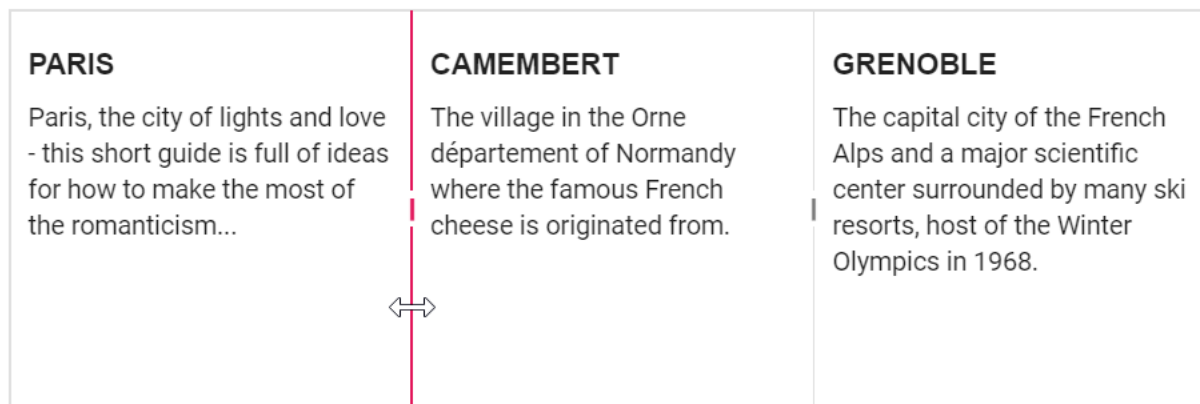
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Min("20%").Max("40%").Content("<div
class='content'><h4 class='h4'>PARIS </h4>Paris, the city of lights and love
- this short guide is full of ideas for how to make the most of the
romanticism...</div>").Add();
 item.Size("200px").Min("30%").Max("60%").Content("<div
class='content'><h4 class='h4'>CAMEMBERT </h4>The village in the Orne
département of Normandy where the famous French cheese is originated
from.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4
class='h4'>GRENOBLE </h4>The capital city of the French Alps and a major
scientific center surrounded by many ski resorts, host of the Winter
Olympics in 1968.</div>").Add();
}).Render()
</div>
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 font-size: 15px;
 }
 .control_wrapper .e-splitter .e-split-bar .e-resize-handler:before {
 content: "\e934";
 font-family: 'e-icons';
 font-size: 11px;
 transform: rotate(90deg);
 }
 .e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-bar,
```

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-resizable-split-
bar::after {
 cursor: e-resize;
}
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.



See Also

- [Resizable split panes](#)

### Pane content in Splitter Control

This section explains how to provide plain text content or HTML markup or integrate other Syncfusion ASP.NET MVC UI controls as content of splitter.

#### HTML Markup

Splitter is a layout based container control. You can render the pane contents from existing HTML markups. Converting HTML markup as splitter pane is easy way to add the panes content dynamically.

#### CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'><h4 class='h4'>PARIS
</h4>Paris, the city of lights and love - this short guide is full of ideas
for how to make the most of the romanticism...</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>CAMEMBERT
</h4>The village in the Orne département of Normandy where the famous French
cheese is originated from.</div>").Add();
})
```

```

 item.Size("200px").Content("<div class='content'><h4 class='h4'>GRENOBLE
</h4>The capital city of the French Alps and a major scientific center
surrounded by many ski resorts, host of the Winter Olympics in
1968.</div>").Add();
 }).Render()
<style>
 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 }
</style>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.

| PARIS                                                                                                                          | CAMEMBERT                                                                                                   | GRENOBLE                                                                                                                                           |
|--------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| Paris, the city of lights and love<br>- this short guide is full of ideas<br>for how to make the most of<br>the romanticism... | The village in the Orne<br>département of Normandy<br>where the famous French<br>cheese is originated from. | The capital city of the French<br>Alps and a major scientific<br>center surrounded by many ski<br>resorts, host of the Winter<br>Olympics in 1968. |

### Syncfusion ASP.NET MVC UI controls

You can render any Syncfusion ASP.NET MVC UI controls along with their native and control events within splitter as pane content.

You can refer [Accordion within splitter](#) and [Listview within splitter](#) samples.

### Plain content

You can add the plain text as a pane contents using either inner HTML or [Content](#) API.

### CSHTML

```

@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'>Left pane</div>").Add();
 item.Size("200px").Content("<div class='content'>Middle
pane</div>").Add();
}

```



```

 item.Size("200px").Content("<div class='content'>Right
pane</div>").Add();
 }).Render()
<style>
 .content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: grid;
 height: 100%;
 }
</style>

```

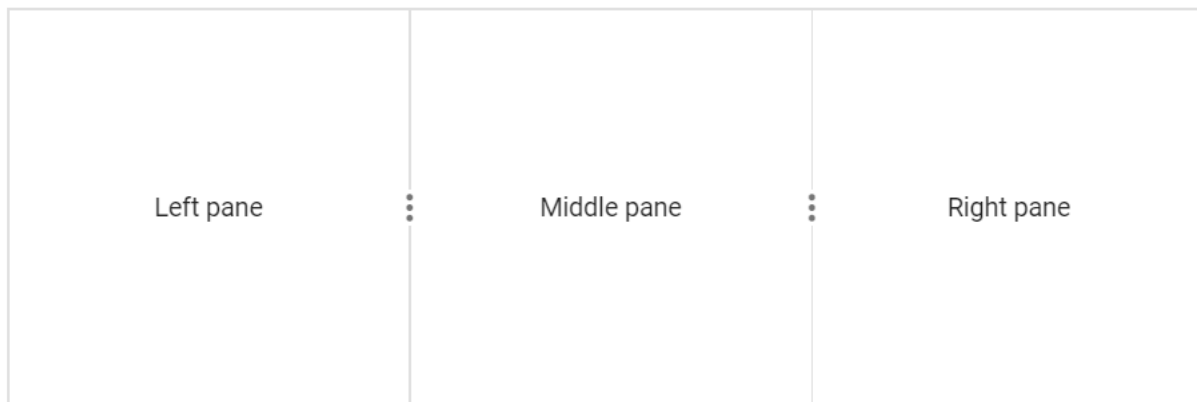
### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.



### Pane content using selector

You can set HTML element as pane content, using the query selectors such as ID or CSS class selectors.

The following code demonstrates how to fetch an element from the document and load it to pane using its ID.

### CSHTML

```

@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Min("60px").Content("#left-pane-content").Add();
 item.Size("200px").Min("60px").Content("#middle-pane-content").Add();
 item.Size("200px").Min("60px").Content("#last-pane-content").Add();
}).Render()
<!-- pane contents -->
<div id="left-pane-content" style="display: none;">

```

```
<div>Left pane<div id='panetext'>size: 25%</div>
 <div id='panetext'>min: 60px</div>
</div>
<div id="middle-pane-content" style="display: none;">
 Middle pane<div id="panetext">size: 50%</div>
 <div id="panetext">min: 60px</div>

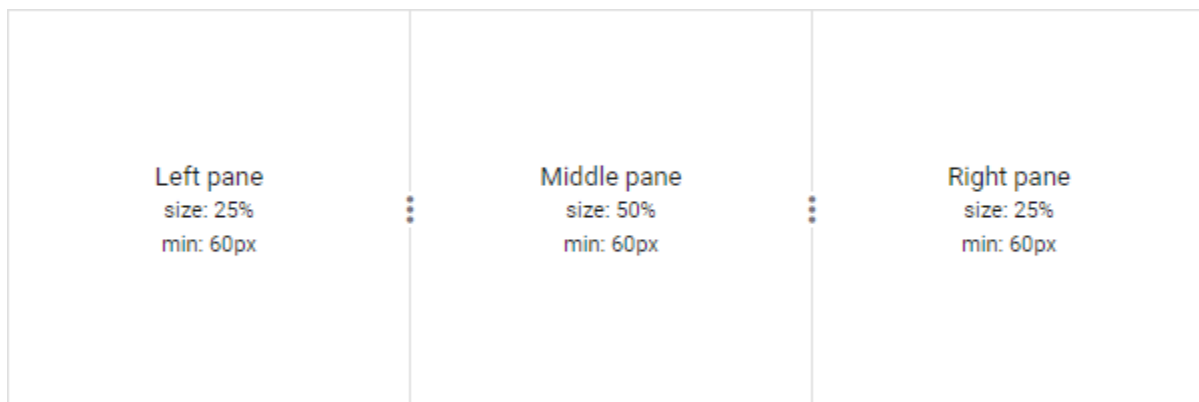
</div>
<div id="last-pane-content" style="display: none;">
 Right pane<div id="panetext">size: 25%</div>
 <div id="panetext">min: 60px</div>

</div>
<style>
#left-pane-content,
#middle-pane-content,
#last-pane-content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: flex;
 height: 100%;
}
#panetext {
 font-size: 11px;
}
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.



## Pane sizing

Splitter allows you to provide pane sizes either in pixel or percentage formats.

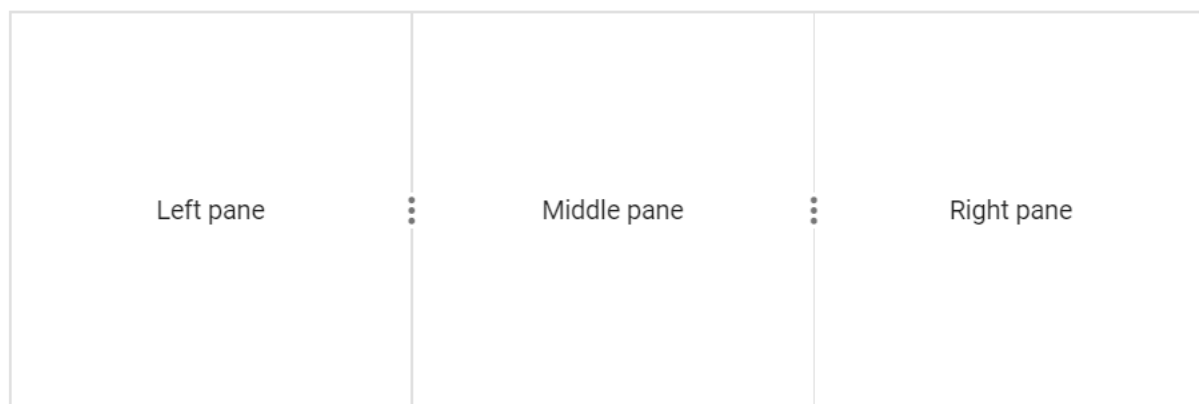
### CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Content("<div class='content'>Left pane</div>").Add();
 item.Size("200px").Content("<div class='content'>Middle
pane</div>").Add();
 item.Size("200px").Content("<div class='content'>Right
pane</div>").Add();
}).Render()
<style>
 .content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: grid;
 height: 100%;
 }
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.



### CSHTML

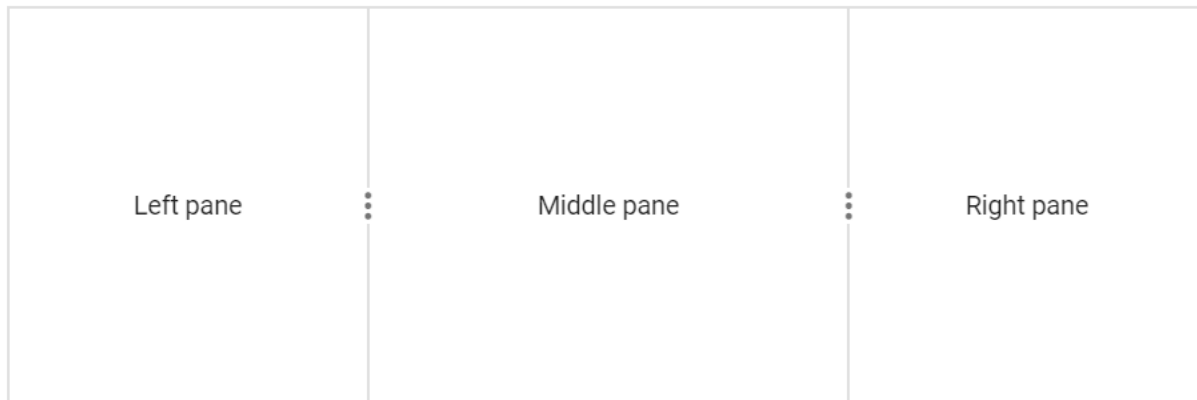
```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("30%").Content("<div class='content'>Left pane</div>").Add();
 item.Size("40%").Content("<div class='content'>Middle pane</div>").Add();
 item.Size("30%").Content("<div class='content'>Right pane</div>").Add();
})
```

```
}).Render()
<style>
 .content {
 text-align: center;
 align-items: center;
 justify-content: center;
 display: grid;
 height: 100%;
 }
</style>
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.



## Auto size panes

You can render the split panes without providing the size values. It will split up the sizes automatically.

## CSHTML

```
@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Content("<div class='content'><h4 class='h4'>Grid</h4>The ASP.NET
DataGrid control, or DataTable is a feature-rich control used to display
data in a tabular format.</div>").Add();
 item.Content("<div class='content'><h4 class='h4'>Schedule </h4>The
ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar
features, thus allowing users to manage their time
efficiently.</div>").Add();
 item.Content("<div class='content'><h4 class='h4'>Chart </h4>ASP.NET
charts, a well-crafted easy-to-use charting package, is used to add
beautiful charts in web and mobile applications").Add();
}).Render()
<style>
```

```

 .content {
 padding: 9px;
 }
 .h4 {
 font-weight: 550;
 }
 }
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.

| Grid                                                                                                           | Schedule                                                                                                                                      | Chart                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format. | The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently. | ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications |

## Fixed pane

You can render the split panes with fixed sizes. Since last pane is a flexible pane, fixed size will not be applied.

## CSHTML

```

@Html.EJS().Splitter("splitter").Width("600px").Height("200px").PaneSettings
(item => {
 item.Size("200px").Resizable(false).Content("<div class='content'><h4
class='h4'>Grid</h4>The ASP.NET DataGrid control, or DataTable is a feature-
rich control used to display data in a tabular format.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Schedule
</h4>The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all
calendar features, thus allowing users to manage their time
efficiently.</div>").Add();
 item.Size("200px").Content("<div class='content'><h4 class='h4'>Chart
</h4>ASP.NET charts, a well-crafted easy-to-use charting package, is used to
add beautiful charts in web and mobile applications").Add();
}).Render()
<style>
 .content {

```

```
padding: 9px;
}
.h4 {
font-weight: 550;
}
</style>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

Output be like the below.

| Grid                                                                                                           | Schedule                                                                                                                                      | Chart                                                                                                                       |
|----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| The ASP.NET DataGrid control, or DataTable is a feature-rich control used to display data in a tabular format. | The ASP.NET Scheduler, a.k.a. event calendar, facilitates almost all calendar features, thus allowing users to manage their time efficiently. | ASP.NET charts, a well-crafted easy-to-use charting package, is used to add beautiful charts in web and mobile applications |

### Different layouts

By using splitter control, you can create the different layouts with multiple and nested panes.

#### Code editor style layout

##### Step 1:

Create the element with two child to render the outer splitter.

```
`html
```

```
@using Syncfusion.EJ2.Layouts
```

```
<div id="target" class="control_wrapper">
```

```
<div>
```

```
@Html.EJS().Splitter("outerSplitter").Created("onCreate").PaneSettings(item => {
```

```
item.Size("53%").Min("30%").Add();
```

```
item.Content(@"<div id='splitter2'><h3 class='h3' style='padding: 5px'>Preview of sample</h3> <div class='splitter-image'> <img class='img1'
```

```
src='https://ej2.syncfusion.com/demos/src/listview/images/albert.png' style='width: 20%;margin: 0 auto;'></div>").Add();
```

```
}).Height("400px").Width("100%).Orientation(Orientation.Vertical).Render()
```

```
</div>
```

```
</div>
```

```
,
```

## Step 2 :

Render the first pane of vertical splitter as a horizontal splitter.

```
`javascript
```

```
<script>
```

```
var HTMLContent = "<div><div class='content'><h3 class='h3'>HTML</h3><div class='code-
preview'><!DOCTYPE html><div><html></div>" +
```

```
"<div><body></div><div id='custom-image'><div style='margin-left:
5px'><img src='src/albert.png'></div>" +
```

```
"<div><div></div><div></body></div><div></html></div>
</div></div></div>";
```

```
var CSSContent = "<div><div class='content'><h3 class='h3'>CSS</h3><div class='code-
preview'>img {<div id='code-text'>margin:0 auto;</div>" +
```

```
"<div id='code-text'>display:flex;</div><div id='code-
text'>height:70px;</div>}</div></div></div>";
```

```
var javaScriptContent = "<div><div class='content'><h3 class='h3'>JavaScript</h3><div class='code-
preview'>var " +
```

```
"image = document.getElementById('custom-image');<div>image.addEventListener('click', function()
{</div>" +
```

```
"<div style='padding-left: 20px;'>// Code block for click action</div>
</div></div></div>";
```

```
function onCreate(args) {
```

```
document.getElementById('outerSplitter').querySelector('.e-pane-vertical').setAttribute('id',
'Innersplitter');
```

```
var splitObj1 = new ej.layouts.Splitter({
```

```
height: '220px',
```

```
paneSettings: [
```

```
{ size: '29%', min: '23%', content: HTMLContent },
```

```
{ size: '20%', min: '15%', content: CSSContent },
```

```
{ size: '35%', min: '35%', content: javaScriptContent }
```

```
],
```

```
width: '100%'
});
splitObj1.appendTo('#Innersplitter');
}
</script>
`
`css
<style>
code-text {
margin-left: 5px;
}
.code-preview {
margin-top: 15px;
font-size: 12px;
}
target {
margin: 20px auto;
max-width: 600px;
}
.control-section {
min-height: 370px;
margin-bottom: 15px;
margin-top: 10px;
}
.h3 {
font-size: 14px;
margin: 4px;
}
.content {
padding: 12px;
}
.splitter-image {
margin: 0 auto;
display: flex;
```



```
height: 115px;
margin-top: 10px;
}
</style>
`
```

Once the above configurations has been completed, you will get the output like [this](#)

Outlook style layout

### Step 1:

Create the element with three panes and place the elements within the pane to render **TreeView**, **ListView** and **RichTextEditor**.

```
`html
<div id="target">
<div>
@Html.EJS().Splitter("splitter").Created("onCreate").PaneSettings(item => {
item.Size("28%").Min("27%").Content(@"<div><div class='content'><div
id='tree'></div></div></div>").Add();
item.Size("33%").Min("23%").Content(@"
").Add(); item.Size("37%").Min("30%").Content(@"
To...
Cc...
Subject
Send Discard
").Add();
}).Height("493px").Width("100%").Render()
</div>
</div>
`
```

### Step 2 :

Place the template script to render the treeview.

```
`javascript
<script id="treeTemplate" type="text/x-template">
<div>
<div class="treeviewdiv">
```

```

<div style="float:left">
${name}
</div>
${if(count)}
<div style="margin-right: 5px; float:right">
${count}
</div>
${/if}
</div>
</div>
</script>
`

```

**Step 3 :**

Render the listed controls one by one.

```

`javascript
<script>
function onCreate(args) {
var inputobj1 = new ej.inputs.TextBox({});
inputobj1.appendTo('#firstname');
var inputobj2 = new ej.inputs.TextBox({});
inputobj2.appendTo('#lastname');
var inputobj3 = new ej.inputs.TextBox({});
inputobj3.appendTo('#subject');
// Data source for TreeView component
var mailBox = [
{ id: 1, name: 'Favorites', hasChild: true },
{ id: 2, pid: 1, name: 'Sales Reports', count: '4' },
{ id: 3, pid: 1, name: 'Sent Items' },
{ id: 4, pid: 1, name: 'Marketing Reports ', count: '6' },
{ id: 5, name: 'Andrew Fuller', hasChild: true, expanded: true },
{ id: 6, pid: 5, name: 'Inbox', selected: true, count: '20' },
{ id: 7, pid: 5, name: 'Drafts', count: '5' },
{ id: 15, pid: 5, name: 'Archive' },

```

```

{ id: 8, pid: 5, name: 'Deleted Items' },
{ id: 9, pid: 5, name: 'Sent Items' },
{ id: 10, pid: 5, name: 'Sales Reports', count: '4' },
{ id: 11, pid: 5, name: 'Marketing Reports', count: '6' },
{ id: 12, pid: 5, name: 'Outbox' },
{ id: 13, pid: 5, name: 'Junk Email' },
{ id: 14, pid: 5, name: 'RSS Feed' },
{ id: 15, pid: 5, name: 'Trash' }
];

// Render the TreeView using template option
var treeObj = new ej.navigations.TreeView({
 fields: { dataSource: mailBox, id: 'id', parentID: 'pid', text: 'name', hasChildren: 'hasChild' },
 nodeTemplate: '#treeTemplate',
});
treeObj.appendTo('#tree');

// tslint:disable:max-line-length
//Define an array of JSON data
var dataSource = [
 { Name: 'Selma Tally', content: '<div><div class="status">Apology marketing email</div><div id="list-message">Hello Ananya Singleton</div>', id: '1', order: 0 },
 { Name: 'Illa Russo', content: '<div><div class="status">Annual conference</div><div id="list-message">Hi jeani Moresa</div></div>', id: '4', order: 0 },
 { Name: 'Camden Macmellon', content: '<div><div class="status">Reference request- Camden hester</div><div id="list-message">Hello Kerry Best,</div></div>', order: 0 },
 { Name: 'Garth Owen', content: '<div><div class="status">Application for job Title</div><div id="list-message">Hello Illa Russo</div></div>', id: '2', order: 0 },
 { Name: 'Ursula Patterson', content: '<div><div class="status">Programmaer Position Applicant</div><div id="list-message">Hello Kerry best,</div></div>', id: '2', order: 0 }
];

// Initialize ListView component
var listObj = new ej.lists.ListView({
 //Set defined data to dataSource property
 dataSource: dataSource,
 cssClass: 'e-list-template',
 //Map the appropriate columns to fields property

```

```

fields: { text: 'Name', groupBy: 'order' },
//Set customized group-header template
groupTemplate: '<div class="e-list-wrapper"></div>',
//Set customized list template
template: '<div class="settings e-list-wrapper e-list-multi-line">' +
'${Name}' +
'${content}' +
'</div>'
});
//Render initialized ListView component
listObj.appendTo('#groupedList');
var button1 = new ej.buttons.Button({ isPrimary: true });
button1.appendTo('#rteSubmit');
var button2 = new ej.buttons.Button();
button2.appendTo('#rteCancel');
var defaultRTE = new ej.richtexteditor.RichTextEditor({ height: '262px' });
defaultRTE.appendTo('#defaultRTE');
}
</script>
`css
<style>
discard {
margin-left: 7px;
}
table {
width: 100%;
}
target {
margin: 20px auto;
max-width: 820px;
}
td {
padding: 2px;

```

Splitter

groupedList.e-listview .e-list-group-item {

```
}
.control-section{
min-height: 370px;
}
.e-treeview .e-list-text {
width: 100%;
}
groupedList.e-listview .e-list-group-item {
height: 0;
}
splitter1 .settings.e-list-wrapper.e-list-multi-line.e-list-avatar {
padding: 15px;
}
buttonSection {
padding: 7px;
}
createpostholder {
padding-left: 3px;
padding-right: 4px;
}
</style>
,
```

Once the above configurations has been completed, you will get the output like [this](#).

See Also

[Multiple panes in Splitter](#)

### CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the split bar

Use the following CSS to customize the split bar properties.

##### *Horizontal split bar*

`CSS

*/ default split bar color /*

```
.e-splitter .e-split-bar.e-split-bar-horizontal{
background: blue;
```

```

}
/ split bar color in hover and active state /
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active {
background: green;
}
`

```

#### Vertical split bar

```

`CSS
/ default split bar color /
.e-splitter .e-split-bar.e-split-bar-vertical {
background: blue;
}
/ split bar color in hover and active state /
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover,
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active {
background: green;
}
`

```

#### Customizing the split bar resize handle

Use the following CSS to customize the split bar resize handle.

#### Horizontal split bar resize handle

```

`CSS
/ default split bar resize handle color /
.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
/ default split bar resize handle color in hover and active state /
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-resize-handler,
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-resize-handler{
color: green;
}
`

```

*Vertical split bar resize handle*``CSS`*/ default split bar resize handle color /*

```
.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
color: rgba(20, 27, 233, 0.54);
}
```

*/ default split bar resize handle color in hover and active state /*

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-resize-handler,
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-resize-handler{
color: green;
}
```

`,`

*Customizing the split bar arrows*

Use the following CSS to customize the split bar arrows.

*Horizontal split bar resize arrows*``CSS`*/ split bar arrows /*

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left::after, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-left::after, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::before, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right::after, .e-
splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-active .e-navigate-arrow.e-arrow-right::after {
background-color: green;
}
```

*/ split bar arrows - circular border /*

```
.e-splitter .e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-left, .e-splitter
.e-split-bar.e-split-bar-horizontal.e-split-bar-hover .e-navigate-arrow.e-arrow-right {
border-color: rgba(33, 227, 22, 0.5);
}
```

`,`

*Vertical split bar resize arrows*``CSS`*/ split bar arrows /*

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-up::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::before, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down::after, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-active .e-navigate-arrow.e-arrow-down::after {
```

```
background-color: green;
```

```
}
```

*/ split bar arrows - circular border /*

```
.e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-up, .e-splitter .e-split-bar.e-split-bar-vertical.e-split-bar-hover .e-navigate-arrow.e-arrow-down {
```

```
border-color: rgba(33, 227, 22, 0.5);
```

```
}
```

,

### To hide the resize handle in Splitter

Use the following CSS to hide the resize handler in the split bar

#### *Hide the horizontal split bar resize arrow*

`CSS

```
.e-splitter .e-split-bar.e-split-bar-horizontal .e-resize-handler {
```

```
display: none;
```

```
}
```

,

#### *Hide the vertical split bar resize arrow*

`CSS

```
.e-splitter .e-split-bar.e-split-bar-vertical .e-resize-handler {
```

```
display: none;
```

```
}
```

,

## Expand and Collapse

### Collapsible panes

Specifies the direction of the Splitter component using the enableRtl property. For writing systems that require it like Arabic, Hebrew, etc., the direction can be switched to right-to-left.

The following code shows how to enable RTL behavior.

#### **CSHTML**

```
@Html.EJS().Splitter("splitter").EnableRtl(true).Width("500px").Height("200px").PaneSettings(item => {
```



```

 item.Size("200px").Content("<div class='content'>Left pane</div>").Add();
 item.Size("200px").Content("<div class='content'>Middle
pane</div>").Add();
 item.Size("200px").Content("<div class='content'>Right
pane</div>").Add();
 }).Render()

```

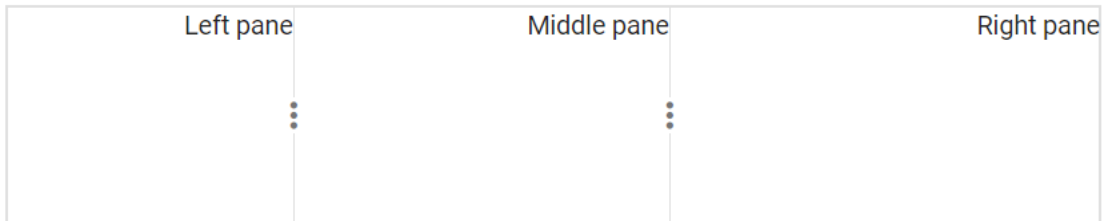
### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Execution of above code's output will be as given below,



See Also

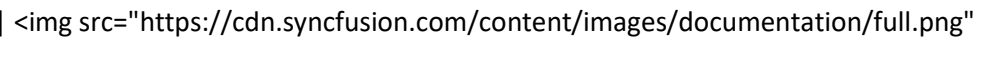
[Multiple panes in Splitter](#)

The Splitter component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Splitter component is outlined below.

| [Accessibility Criteria](#) | [Compatibility](#) |

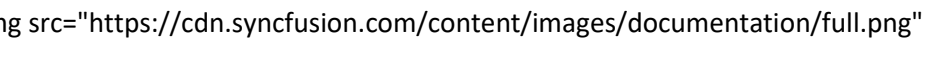
| -- | -- |

| [WCAG 2.2 Support](#) |  alt="Yes" > |

| [Section 508 Support](#) |  alt="Yes" > |

| [Screen Reader Support](#) |  alt="Yes" > |

| [Right-To-Left Support](#) |  alt="Yes" > |

| [Color Contrast](#) |  alt="Yes" > |

```

| Mobile Device Support | |
| Keyboard Navigation Support | |
| Accessibility Checker Validation | |
| Axe-core Accessibility Validation | |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>
<div> - All
features of the component meet the requirement.</div>
<div> - Some features of the component do not meet the requirement.</div>
<div> - The component does not meet the requirement.</div>

```

### Keyboard interaction

You can use the following key shortcuts to access the splitter without interruptions:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| Tab | Helps in focusing the splitter on the page and switching between the consecutive splitter bars. |

| Shift + Tab | Helps in focusing the previous splitter bar element on the splitter. |

| Right arrow | Helps in moving the active horizontal orientated splitter bar to its Right side. |

| Left arrow | Helps in moving the active horizontal orientated splitter bar to its Left side. |

| Up arrow | Helps in moving the active vertical orientated splitter bar to its Up side. |

| Down arrow | Helps in moving the active vertical orientated splitter bar to its Down side. |

| Enter | Helps to toggle between expand and collapse actions of the splitter bar when it is active. |

### Ensuring accessibility

The Splitter component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Splitter component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Splitter component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## Migration from Essential JS 1

This article describes the API migration process of Splitter component from Essential JS 1 to Essential JS 2.

### Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Adding custom class | **Property:** *cssClass* <br /><br />@{Html.EJ().Splitter("splitter")<br />.CssClass("customClass")<br />.Render();} <br /> | **Property:** *cssClass* <br /><br />@Html.EJS().Splitter("splitter")<br />.CssClass("customClass")<br />.Render();<br /> |

| Adjusting Height | **Property:** *height* <br /><br />@{Html.EJ().Splitter("splitter")<br />.Height("100%").Render();} <br /> | **Property:** *height* <br /><br />@Html.EJS().Splitter("splitter")<br />.Height("100%").Render(); <br /> |

| Adjusting Width | **Property:** *width* <br /><br />@{Html.EJ().Splitter("splitter")<br />.Width("600").Render();}<br /> | **Property:** *width* <br /><br />@Html.EJS().Splitter("splitter")<br />.Width("100%").Render();<br /> |

| Orientation | **Property:** *orientation* <br /><br />@{Html.EJ().Splitter("splitter")<br />.Orientation(Orientation.Vertical)<br />.Render();}<br /> | **Property:** *orientation* <br /><br />@Html.EJS().Splitter("splitter")<br />.Orientation<br />(Syncfusion.EJ2.Layouts<br />.Orientation.Vertical)<br />.Render();<br /> |

| Separator Size | Not Available | **Property:** *separatorSize* <br /><br />@Html.EJS().Splitter("splitter")<br />.SeparatorSize(4).Render();<br /> |

| Adding HTML attributes | **Property:** *htmlAttributes* <br /><br />@{<br />IDictionary<string, object><br />htmlAttribute = new Dictionary<br /><string, object>(); <br />htmlAttribute.Add("class",<br />"my-class")<br />}<br /><br />@{Html.EJ().Splitter("splitter")<br />.HtmlAttributes(htmlAttribute)<br />.Render();}<br /> | Not Available |

| Customize expand/collapse icons | **Property:** <br /> *expanderTemplate* <br /><br />@{Html.EJ().Splitter("splitter")<br />.ExpanderTemplate<br /><img class="eimg"<br />src="expander.png"<br />alt="employee"/>}.Render();}<br /> | Not Available |

| Make control flexible for mobile view | **Property:** *isResponsive* <br /><br />@{Html.EJ().Splitter("splitter")<br />.IsResponsive(true).Render();}<br /> | By default, Splitter works with mobile mode. |

| Refresh the Splitter | **Method:** *refresh()* <br /><br />  
 @Html.EJ().Splitter("splitter")<br/>.Render();<br/>\$("#splitter").ejSplitter("refresh")<br/> |  
**Method:** *refresh()* <br /><br />  
 />@Html.EJS().Splitter("splitter")<br/>.Render();<br/>\$("#splitter").ejSplitter("refresh");<br /> |

| Destroy the Control | **Method:** *destroy()* <br /><br />  
 />@Html.EJ().Splitter("splitter")<br/>.Render();<br/>\$("#splitter").ejSplitter("destroy")<br />  
 /> | **Method:** *destroy()* <br /><br />  
 />@Html.EJS().Splitter("splitter")<br/>.Render();<br/>\$("#splitter").ejSplitter("destroy");<br /> |

| Event triggers after the Splitter created successfully | **Event:** *create* <br /><br />  
 />@Html.EJ().Splitter("splitter")<br/>.ClientSideEvents(e=><br/>e.Create("onCreate"))<br/>.Render();<br/>function onCreate(args) {}<br/> | **Event:** *created* <br /><br />  
 />@Html.EJS().Splitter("splitter")<br/>.ClientSideEvents(e=><br/>e.Create("onCreate")).Render();<br/>function onCreate(args) {}<br/> |

| Event triggers when Splitter has been destroyed | **Event:** *destroy* <br /><br />  
 />@Html.EJ().Splitter("splitter")<br/>.ClientSideEvents(e=><br/>e.Destroy("onDestroy"))<br/>.Render();<br/>function onDestroy(args) {}<br /> }); | Not Available |

#### Accessibility and Localization

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Keyboard Navigation | **Property:** *allowKeyboardNavigation* <br /><br />  
 @Html.EJ().Splitter("splitter")<br/>.AllowKeyboardNavigation(true)<br/>.Render();<br/> | No  
 separate property for enable/disable keyboard navigation. Its enabled by default. |

| Right to Left | **Property:** *enableRTL* <br /><br />  
 />@Html.EJ().Splitter("splitter")<br/>.EnableRTL(false).Render();<br/> | **Property:** *enableRtl*<br />  
 /><br />@Html.EJS().Splitter("splitter")<br/>.EnableRtl(true).Render();<br/> |

#### Control State

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Enable/Disable the control | Not Available | **Property:** *enabled* <br /><br />  
 />@Html.EJS().Splitter("splitter")<br/>.Enabled(true).Render();<br/> |

#### State Maintenance

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Save the model value in local storage or cookies | Not Available | **Property:** *enablePersistence* <br />  
 /><br />@Html.EJS().Splitter("splitter")<br/>.EnablePersistence(true).Render();<br/> |

#### Pane Properties

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Property:** *properties* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties().Render();}<br /> | **Property:** *paneSettings* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings().Render();}<br/> |

| Pane Content | Not Available | **Property:** *content* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=>{item.Content("<div>First Pane Content</div>")})<br/>.Render();}<br/> |

| Change the size of the pane | **Property:** *paneSize* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties(p=><br/>{p.Add().PaneSize("30px")})<br/>.Render();}<br /> | **Property:** *size* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=><br/>{item.Size("25%")})<br/>.Render();}<br/> |

| Minimum pane size | **Property:** *minSize* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties(p=>{p.Add()<br/>.MinSize(30)})<br/>.Render();}<br/> | **Property:** *min* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=><br/>{item.Min("60px")})<br/>.Render();}<br/> |

| Maximum pane size | **Property:** *maxSize* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties(p=><br/>{p.Add().MaxSize(30)})<br/>.Render();}<br /> | **Property:** *max* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=><br/>{item.Max("60px")})<br/>.Render();}<br/> |

| Enable/Disable the Pane Resizable behavior | **Property:** *resizable* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties(p=><br/>{p.Add().Resizable(false)})<br/>.Render();}<br /> | **Property:** *resizable* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=><br/>{item.Resizable(false)})<br/>.Render();}<br/> |

| Collapsible | **Property:** *collapsible* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties(p<br/>=>{p.Add().Collapsible(true)})<br/>.Render();}<br /> | **Property:** *collapsible* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=><br/>{item.Collapsible(true)})<br/>.Render();}<br/> |

| Expandable | **Property:** *expandable* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.PaneProperties(p=><br/>{p.Add().Expandable(true)})<br/>.Render();}<br /> | Not Available |

| Collapsed | Not Available | **Property:** *collapsed* <br /><br />@{Html.EJS().Splitter("splitter")<br/>.PaneSettings(item=><br/>{item.Collapsed(true)})<br/>.Render();}<br/> |

| Add Pane | **Method:** *addItem()* <br /><br />@{Html.EJ().Splitter("splitter")<br/>.Render();}<br/>\$("#splitter").ejSplitter<br/>("addItem", "New Pane 0",<br/>{paneSize:20, minSize:20,<br/>maxSize: 100}, 0);<br/> | **Method:** *addPane()* <br /><br />

```

/>@Html.EJS().Splitter("splitter")
.Render();
$("#splitter").ejSplitter
("addPane",
"New Pane 0",
{ size: "25%",
content: "Pane"}, 0);
|

| Remove Pane | Method: removeItem()

@{Html.EJ().Splitter("splitter")
.Render();}
$("#splitter").ejSplitter
("remov
eItem", 0);
| Method: removePane()

@Html.EJS().Splitter("splitter")
.Render();
$("#splitter")
.ejSplitter("removePane
", 0);
|

| Collapse Pane | Method: collapse()

@{Html.EJ().Splitter("splitter")
.Render();}
$("#splitter")
.ejSplitter("collapse",
0);
| Method: collapse()

@Html.EJS().Splitter("splitter")
.Render();
$("#splitter")
.ejSplitter("collapse",
0);
|

| Expand Pane | Method: expand()

@{Html.EJ().Splitter("splitter")
.Render();}
$("#splitter")
.ejSplitter("expand",
0);
| Method: expand()

@Html.EJS().Splitter("splitter")
.Render();
$("#splitter").ejSplitter
("expand",
0);
|

| Event triggers when before panes get expanded/collapsed | Event: beforeExpandCollapse

@{Html.EJ().Splitter("splitter")
.ClientSideEvents(e=>
e.BeforeExpandCollapse
("
onBeforeExpandCollapse"))
.Render();}
function
onBeforeExpandCollapse
(args)
{}
| Event: beforeExpand

@Html.EJS().Splitter("splitter")
.ClientSideEvents(e=>
e.BeforeExpand("onBeforeExp
and"))
.Render();
function onBeforeExpand(args) {}
Event: beforeCollapse
@Html.EJS().Splitter("splitter")
.ClientSideEvents(e=>
e.BeforeCollapse("onBeforeCol
lapse"))
.Render();
function onBeforeCollapse(args) {}
|

| Event triggers when after panes get expanded/collapsed | Event: expandCollapse

@{Html.EJ().Splitter("splitter")
.ClientSideEvents(e=>
e.ExpandCollapse
("onExp
andCollapse"))
.Render();}
function onExpandCollapse (args) {}
| Event: expand

@Html.EJS().Splitter("splitter")
.ClientSideEvents(e=>
e.Expand("onExpand"))
.R
ender();
function onExpand(args) {}
Event: collapse

@Html.EJS().Splitter("splitter")
.ClientSideEvents(e=>
e.Collapse("onCollapse"))
.R
ender();
function onCollapse(args) {}
|

| Event triggers when Resizing the pane | Event: resize

@{Html.EJ().Splitter("splitter")
.ClientSideEvents(e=>
e.Resize("onResize"))
.Ren
der();}
function onResize (args) {}
| Event: resizing

@Html.EJS().Splitter("splitter")
.ClientSideEvents(e=>
e.Resizing("onResizing"))
.
Render();
function onResizing(args) {}
|

| Event triggers when pane is started to resize | Not Available | Event: resizeStart

@Html.EJS().Splitter("splitter")
.ClientSideEvents(e=>
e.ResizeStart("onResizeStart"))

.Render();
function onResizeStart(args) {}
|

```

| Event triggers when pane is stopped to resize | Not Available | **Event:** *resizeStop* <br /><br />@Html.EJS().Splitter("splitter")<br />.ClientSideEvents(e=><br />e.ResizeStop("onResizeStop"))<br />.Render();<br />function onResizeStop(args) {}<br /> |

| Event triggers when click template icon | **Event:** *clickOnExpander* <br /><br />@Html.EJS().Splitter("splitter")<br />.ClientSideEvents(e=>e<br />.ClickOnExpander<br />("onClickOnExpander"))<br />.Render();<br />function clickOnExpander (args) {}<br /> | Not Available |

### Animation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| EnableAnimation | **Property:** *enableAnimation* <br /><br />@Html.EJS().Splitter("splitter")<br />.EnableAnimation(true)<br />.Render();<br /> | Not Available |

| AnimationSpeed | **Property:** *animationSpeed* <br /><br />@Html.EJS().Splitter("splitter")<br />.AnimationSpeed(150)<br />.Render();<br /> | Not Available |

## SpreadSheet

### Overview of the ASP.NET MVC Spreadsheet control

The Spreadsheet is an user interactive control to organize and analyze data in tabular format with configuration options for customization. It will load data by importing an Excel/CSV file or from local and remote data sources such as JSON, RESTful services, OData services, and more. The populated data can be exported as Excel with xlsx, xls, CSV and PDF formats.

### Key features

- [Data sources](#): Bind the Spreadsheet control with an array of objects or data from a web service using `DataManager`.
- [Virtualization](#): Provides the option to load large amount of data without performance degradation.
- [Selection](#): Provides the option to select a cell or range of cells.
- [Editing](#): Provides the option to dynamically edit a cell.
- [Formulas](#): Provides built-in calculation library with pre-defined formulas and named range support.
- [Clipboard](#): Provides the option to perform clipboard operations.
- [Cell formatting](#): Provides the option to customize the appearance of cells.
- [Number formatting](#): Provides the option to format the cell value.
- [Open](#): Provides the option to open Excel and CSV files in Spreadsheet.
- [Save](#): Provides the option to save Spreadsheet data as Excel, CSV, and PDF documents.
- [Sorting](#): Helps you to arrange the data to particular order in a selected range of cells.
- [Filtering](#): Helps you to view specific rows in the Spreadsheet by hiding the other rows.
- [Undo Redo](#): Provides the option to perform undo redo operations in Spreadsheet.
- [Collaborative editing](#): Provides the option for real time changes across multiple users in the Spreadsheet.
- [Hyperlink](#): Provides the option to navigate to web link or cell reference within the sheet or to other sheet in Spreadsheet.

- [Resize](#): Allows you to change the row height and column width. Auto fit the rows and columns based on its content.
- [Wrap text](#): Provides the option to display the large content as multiple lines in a single cell.
- [Data validation](#): Provides the option to validate edited values based on data validation rules defined for a cell or range of cells.
- [Find and replace](#): Provides the option to find the data and replace it across all sheets in Spreadsheet.
- [Protect sheet](#): Provides the option to restrict user actions like cell editing, row and column insertion, deletion, and resizing.
- [Borders](#): Provides the option to customize cell gridlines such as color and its style for enhanced UI.
- [Show/hide](#): Provides the option to show/hide rows, columns and sheets.
- [Insert/delete](#): Provides the option to insert/delete rows, columns and sheets.
- [Merge cells](#): Provides the option to combine two or more cells located in the same row or column into a single cell.
- [Conditional formatting](#): Provides the option to format a cell or range of cells based on conditions applied.
- [Autofill](#): Provides the option to fill or copy a series or pattern of values and formats into adjacent cells in any direction.
- [Clear](#): Provides the option to clear the content, formats, and hyperlinks applied to a cell or range of cells in a Spreadsheet.
- [Aggregates](#): Provides the option to check the sum, average, count, and more for the selected cells or range in the sheet.
- [Picture](#): Allows you to view, insert, and modify a picture in a Spreadsheet with customizing options.
- [Chart](#): Transforms your Spreadsheet data to an intuitive overview for better understanding and to make smart business decisions.
- [Freeze panes](#): Allows you to keep the specified rows and columns always visible at the top and left side of the sheet while scrolling through the sheet.
- [Password protection](#): Allows you to protect the workbook with a password.
- [Multi-line editing](#): Allows you to insert a line break between paragraphs of the text within a cell in a Spreadsheet.
- [Calculate range selection](#): Helps you to select a range or multiple ranges when editing a formula in a cell.
- [Right-to-left \(RTL\)](#): Aligns content in the Spreadsheet control from right to left.
- [Templates](#): Templates can be used to create custom user experiences in the Spreadsheet.
- [Globalization](#): Personalize the Spreadsheet control with different languages, as well as culture-specific number, date, and time formatting.
- [Accessibility](#): Provides with built-in accessibility support which helps to access all the Spreadsheet control features through the keyboard, screen readers, or other assistive technology devices.

## Getting Started with ASP.NET MVC Spreadsheet Control

This section briefly explains about how to include [ASP.NET MVC Spreadsheet](#) control in your ASP.NET MVC application using Visual Studio.

### Prerequisites

#### [System requirements for ASP.NET MVC controls](#)



Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### **~/ \_LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC

controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

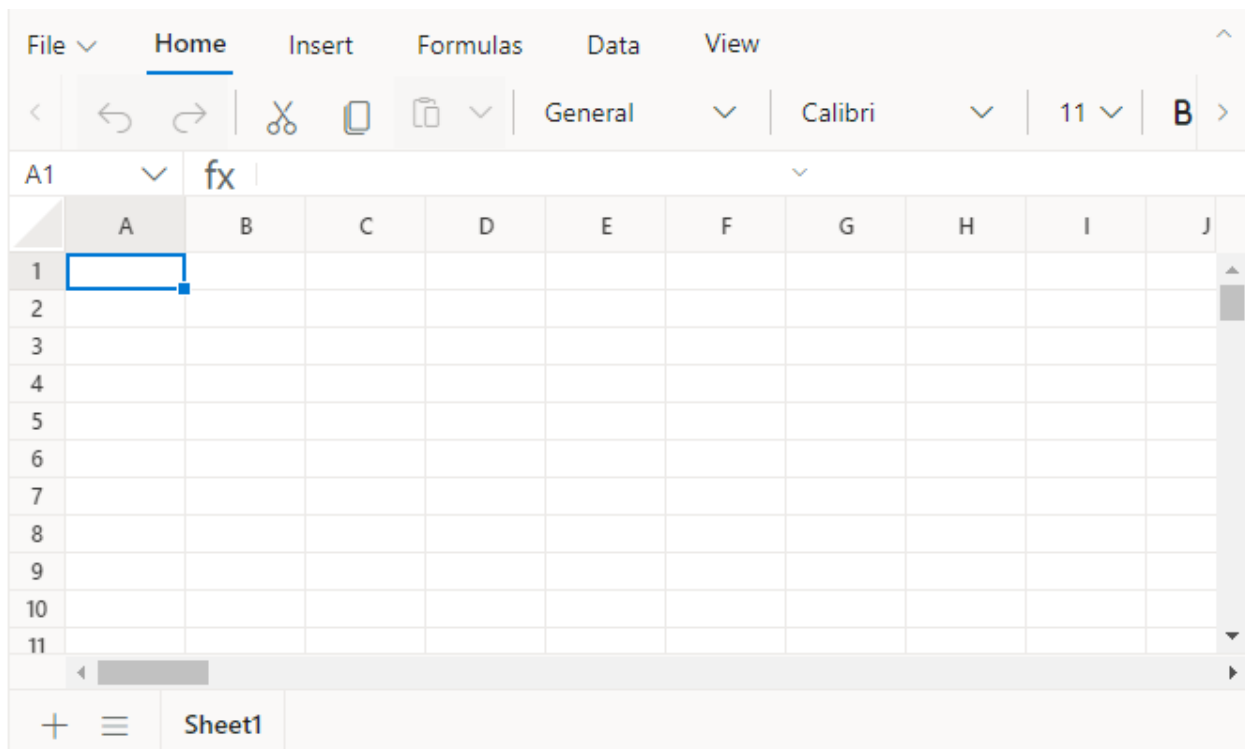
Add ASP.NET MVC Spreadsheet control

Now, add the Syncfusion ASP.NET MVC Spreadsheet control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Spreadsheet control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

See also

- [Data Binding](#)
- [Open and Save](#)

**Note:** You can refer to our [ASP.NET MVC Spreadsheet](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Spreadsheet example](#) that shows you how present and manipulate data, including editing, formulas, formatting, importing, and exporting.

### Open and Save in Spreadsheet control

To import an excel file, it needs to be read and converted to client side Spreadsheet model. The converted client side Spreadsheet model is sent as JSON which is used to render Spreadsheet. Similarly, when you save the Spreadsheet, the client Spreadsheet model is sent to the server as JSON for processing and saved. Server configuration is used for this process.

#### Open

The Spreadsheet component opens an Excel document with its data, style, format, and more. To enable this feature, set [allowOpen](#) as `true` and assign service url to the [openUrl](#) property.

#### User Interface:

In user interface you can open an Excel document by clicking **File > Open** menu item in ribbon.

The following sample shows the **Open** option by using the [openUrl](#) property in the Spreadsheet control. You can also use the [beforeOpen](#) event to trigger before opening an Excel file.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").OpenUrl("Open").AllowOpen(true).BeforeOpen("beforeOpen").Render()
<script>
 function beforeOpen(args) {
 // your code snippets here
 }
</script>
```

#### OPENCONTROLLER.CS

```
public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}
```

Find the below table for the beforeOpen event arguments.

| Parameter | Type | Description |
|-----------|------|-------------|
|-----------|------|-------------|

|       |       |       |
|-------|-------|-------|
| ----- | ----- | ----- |
|-------|-------|-------|

|      |                            |                                                                                                                                                 |
|------|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| file | FileList or string or File | To get the file stream. <b>FileList</b> - contains length and item index. <br/><br><b>File</b> - specifies the file lastModified and file name. |
|------|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|

| cancel | boolean | To prevent the open operation. |

| requestData | object | To provide the Form data. |

**Note:** \* Use **Ctrl + O** keyboard shortcut to open Excel documents.

<br/> \* The default value of the [allowOpen](#) property is true. For demonstration purpose, we have showcased the [allowOpen](#) property in previous code snippet.

#### *Open an external URL excel file while initial load*

You can achieve to access the remote excel file by using the [created](#) event. In this event you can fetch the excel file and convert it to a blob. Convert this blob to a file and **open** this file by using Spreadsheet component open method.

#### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").OpenUrl("Open").AllowOpen(true).Created("created").Render()
<script>
 function created() {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 fetch("https://cdn.syncfusion.com/scripts/spreadsheet/Sample.xlsx")
 // fetch the remote url
 .then((response) => {
 response.blob().then((fileBlob) => { // convert the excel
file to blob
 var file = new File([fileBlob], "Sample.xlsx");
 //convert the blob into file
 spreadsheet.open({ file: file }); // open the file into
Spreadsheet
 })
 })
 }
</script>
```

#### **OPENCONTROLLER.CS**

```
public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}
```

#### *To add custom header during open*

You can add your own custom header to the open action in the Spreadsheet. For processing the data, it has to be sent from server to client side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the [beforeOpen](#) event, the custom header can be added to the request during open action.

#### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").OpenUrl("Home/Open").AllowOpen(true).BeforeOpen("beforeOpen").Render()
<script>
```

```
function beforeOpen(args) {
 args.requestData["headers"] = {
 Authorization: "YOUR TEXT"
 };
}
</script>
```

### OPENCONTROLLER.CS

```
public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}
```

### *Open excel file into a read-only mode*

You can open excel file into a read-only mode by using the [openComplete](#) event. In this event, you must protect all the sheets and lock its used range cells by using [protectSheet](#) and [lockCells](#) methods.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").OpenUrl("Home/Open").OpenComplete("openComplete").Render()
<script>

 function openComplete(args) {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var sheets = spreadsheet.sheets;
 for (var index = 0; index < sheets.length; index++) {
 var name = spreadsheet.sheets[index].name;
 var protectSetting = {
 selectCells: true,
 formatCells: false,
 };
 //To protect the sheet using sheet name
 spreadsheet.protectSheet(name, protectSetting);
 var address = ej.spreadsheet.getRangeAddress([
 0,
 0,
 sheets[index].usedRange.rowIndex,
 sheets[index].usedRange.colIndex,
]);
 //To lock the used range cells
 spreadsheet.lockCells(name + '!' + address, true);
 }
 }
</script>
```

### OPENCONTROLLER.CS

```
public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
```

```
open.File = openRequest.Files[0];
return Content(Workbook.Open(open));
}
```

#### External workbook confirmation dialog

When you open an excel file that contains external workbook references, you will see a confirmation dialog. This dialog allows you to either continue with the file opening or cancel the operation. This confirmation dialog will appear only if you set the `AllowExternalWorkbook` property value to **false** during the open request, as shown below. This prevents the spreadsheet from displaying inconsistent data.

```
`csharp
public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.AllowExternalWorkbook = false;
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}
```

This feature is only applicable when importing an Excel file and not when loading JSON data or binding cell data.

![External workbook confirmation dialog](./images/external-reference-dialog-alert%20.png)

#### Supported file formats

The following list of Excel file formats are supported in Spreadsheet:

- Microsoft Excel (.xlsx)
- Microsoft Excel 97-2003 (.xls)
- Comma Separated Values (.csv)
- Excel Macro-Enabled Workbook (.xlsm)
- Excel Binary Workbook(.xlsb)

#### Save

The Spreadsheet component saves its data, style, format, and more as Excel file document. To enable this feature, set `allowSave` as **true** and assign service url to the `saveUrl` property.

#### User Interface:

In user interface, you can save Spreadsheet data as Excel document by clicking **File > Save As** menu item in ribbon.

The following sample shows the **Save** option by using the `saveUrl` property in the Spreadsheet control. You can also use the `beforeSave` event to trigger before saving the Spreadsheet as an Excel file.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").SaveUrl("Save").AllowSave(true).Render();
```

### SAVECONTROLLER.CS

```
public void Save(SaveSettings saveSettings)
{
 Workbook.Save(saveSettings);
}
```

Find the below table for the beforeSave event arguments.

| Parameter    | Type     | Description                                                                                                                                                              |
|--------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| url          | string   | Specifies the save url.                                                                                                                                                  |
| fileName     | string   | Specifies the file name.                                                                                                                                                 |
| saveType     | SaveType | Specifies the saveType like Xlsx, Xls, Csv and Pdf.                                                                                                                      |
| customParams | object   | Passing the custom parameters from client to server while performing save operation.                                                                                     |
| isFullPost   | boolean  | It sends the form data from client to server, when set to true. It fetches the data from client to server and returns the data from server to client, when set to false. |
| needBlobData | boolean  | You can get the blob data if set to true.                                                                                                                                |
| cancel       | boolean  | To prevent the save operations.                                                                                                                                          |

**Note:** \* Use **Ctrl + S** keyboard shortcut to save the Spreadsheet data as Excel file.

\* The default value of [allowSave](#) property is **true**. For demonstration purpose, we have showcased the [allowSave](#) property in previous code snippet.

\* Demo purpose only, we have used the online web service url link.

*To send and receive custom params from client to server*

Passing the custom parameters from client to server by using [beforeSave](#) event.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").SaveUrl("Save").AllowSave(true).BeforeSave("beforeSave").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<script>
 function beforeSave(args) {
 args.customParams = { customParams: 'you can pass custom params in server side' }; // you can pass the custom params
 }
</script>
```

**CUSTOMPARAMSCONTROLLER.CS**

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

public void Save(SaveSettings saveSettings, string customParams)
{

```



```
Workbook.Save(saveSettings);
}
```

### *To add custom header during save*

You can add your own custom header to the save action in the Spreadsheet. For processing the data, it has to be sent from client to server side and adding customer header can provide privacy to the data with the help of Authorization Token. Through the [fileMenuItemSelect](#) event, the custom header can be added to the request during save action.

### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").SaveUrl("Save").AllowSave(true).FileMenuItemSelect("fileMenuItemSelect").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<script>
function fileMenuItemSelect(args) {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (args.item.text === "Microsoft Excel") {
 args.cancel = true;
 spreadsheet.saveAsJson().then(response => {
 var formData = new FormData();
 formData.append(
 "JSONData",
 JSON.stringify(response.jsonObject.Workbook)
);
 formData.append("fileName", "Sample");
 formData.append("saveType", "Xlsx");
 fetch(
 "https://services.syncfusion.com/aspnet/production/api/spreadsheet/save",
 {
 method: "POST",
 headers: { Authorization: "YOUR TEXT" },
 body: formData
 }
).then(response => {
 response.blob().then(data => {
 var anchor = createElement("a", {
 attrs: { download: "Sample.xlsx" }
 });
 var url = URL.createObjectURL(data);
 anchor.href = url;
 document.body.appendChild(anchor);
 anchor.click();
 URL.revokeObjectURL(url);
 document.body.removeChild(anchor);
 });
 });
 });
 }
}
```

```
}
</script>
```

**CUSTOMHEADERCONTROLLER.CS**

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}
```

```
public void Save(SaveSettings saveSettings, string customParams)
{
 Workbook.Save(saveSettings);
}
```

#### To change the PDF orientation

By default, the PDF document is created in **Portrait** orientation. You can change the orientation of the PDF document by using the `args.pdfLayoutSettings.orientation` argument settings in the [beforeSave](#) event.

The possible values are:

- **Portrait** - Used to display content in a vertical layout.
- **Landscape** - Used to display content in a horizontal layout.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").SaveUrl("Save").AllowSave(true).BeforeSave("beforeSave").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<script>
 function beforeSave(args) {
 args.pdfLayoutSettings.orientation = 'Landscape'; // You can change
 the orientation of the PDF document
 }
</script>
```

#### PDFORIENTATIONCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
 Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
 "07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
 Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
 Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
 Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
 "09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
 "Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
 Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
 Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
 Amount= "9578.45" },
 }
```

```

 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

public void Save(SaveSettings saveSettings, string customParams)
{
 Workbook.Save(saveSettings);
}

```

### Supported file formats

The following list of Excel file formats are supported in Spreadsheet:

- Microsoft Excel (.xlsx)
- Microsoft Excel 97-2003 (.xls)
- Comma Separated Values (.csv)
- Portable Document Format (.pdf)

### Methods

To save the Spreadsheet document as an **xlsx**, **xls**, **csv**, or **pdf** file, by using **save** method should be called with the **url**, **fileName** and **saveType** as parameters. The following code example shows to save the spreadsheet file as an **xlsx**, **xls**, **csv**, or **pdf** in the button click event.

### CSHTML

```

@Html.EJS().DropDownButton("element").Content("Save").Items((IEnumerable<object>)ViewBag.items).Select("itemSelect").Render()

```

```

@Html.EJS().Spreadsheet("spreadsheet").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
 function itemSelect(args) {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (args.item.text === 'Save As xlsx')
 spreadsheetObj.save({url:
'https://services.syncfusion.com/aspnet/production/api/spreadsheet/save',
fileName: "Sample", saveType: "Xlsx"});
 if (args.item.text === 'Save As xls')
 spreadsheetObj.save({url:
'https://services.syncfusion.com/aspnet/production/api/spreadsheet/save',
fileName: "Sample", saveType: "Xls"});
 if (args.item.text === 'Save As csv')
 spreadsheetObj.save({url:
'https://services.syncfusion.com/aspnet/production/api/spreadsheet/save',fil
eName: "Sample", saveType: "Csv"});
 if (args.item.text === 'Save As pdf')
 spreadsheetObj.save({url:
'https://services.syncfusion.com/aspnet/production/api/spreadsheet/save',fil
eName: "Sample", saveType: "Pdf"});
 }
</script>

```

### OPENSOURCECONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 }
}

```

```

 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Save As xlsx"
 });
 items.Add(new
 {
 text = "Save As xls"
 });
 items.Add(new
 {
 text = "Save As csv"
 });
 items.Add(new
 {
 text = "Save As pdf"
 });
 ViewBag.items = items;
 ViewBag.DefaultData = data;
 return View();
}

```

### Server Configuration

In Spreadsheet component, import and export operation processed in **server-side**, to use importing and exporting in your projects, it is required to create a server with any of the following web services.

- WebAPI
- WCF Service
- ASP.NET MVC Controller Action

**Note:** \* Refer the above open and save operation to shows the create a server using WebAPI configuration for Excel import and export. In ASP.NET Core and ASP.NET MVC you can configure the server in controller.

### Server Dependencies

Open and save helper functions are shipped in the Syncfusion.EJ2.Spreadsheet package, which is available in Essential Studio and [nuget.org](https://www.nuget.org). Following list of dependencies required for Spreadsheet open and save operations.

#### | Platforms | Assembly | Nuget Package |

| ---- | ---- | ---- |

| ASP.NET Core (Targeting .NET Core) | Syncfusion.EJ2.AspNet.Core <br/>  
 Syncfusion.EJ2.Spreadsheet.AspNet.Core <br/> Syncfusion.Compression.Net.Core <br/>  
 Syncfusion.XlsIO.Net.Core <br/> Syncfusion.XlsIORenderer.Net.Core <br/> |  
[Syncfusion.EJ2.Spreadsheet.AspNet.Core](#) <br/> [Syncfusion.XlsIORenderer.Net.Core](#) |

| ASP.NET MVC4 | Syncfusion.EJ2.MVC4 <br/> Syncfusion.EJ2.Spreadsheet.AspNet.MVC4 <br/>  
 Syncfusion.Compression.Base <br/> Syncfusion.XlsIO.AspNet.Mvc4 <br/>  
 Syncfusion.ExcelToPdfConverter.AspNet.Mvc4 <br/> | [Syncfusion.EJ2.Spreadsheet.AspNet.MVC4](#) <br/>  
[Syncfusion.ExcelToPdfConverter.AspNet.Mvc4](#) |

| ASP.NET MVC5 | Syncfusion.EJ2.MVC5 <br/> Syncfusion.EJ2.Spreadsheet.AspNet.MVC5 <br/>  
 Syncfusion.Compression.Base <br/> Syncfusion.XlsIO.AspNet.Mvc5 <br/>  
 Syncfusion.ExcelToPdfConverter.AspNet.Mvc5 <br/> | [Syncfusion.EJ2.Spreadsheet.AspNet.MVC5](#) <br/>  
[Syncfusion.ExcelToPdfConverter.AspNet.Mvc5](#) |

See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

### Worksheet in Spreadsheet control

Worksheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data.

#### Add sheet

You can dynamically add or insert a sheet by one of the following ways,

- Click the **Add Sheet** button in the sheet tab. This will add a new empty sheet next to current active sheet.
- Right-click on the sheet tab, and then select **Insert** option from the context menu to insert a new empty sheet before the current active sheet.
- Using **insertSheet** method, you can insert one or more sheets at your desired index.

The following code example shows the insert sheet operation in spreadsheet.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowRibbon(false).ShowFormulaBar(false).Created("created").Sheets(sheet =>
```

```

{
 sheet.Name("Shipment Details").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();

 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
function created() {
 // Applies style formatting to the active sheet before inserting a new
 sheet
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:H1');
 this.cellFormat({ textAlign: 'center' }, 'D2:H11');
 // inserting a new sheet with data at 1st index
 // You can also insert empty sheets by specifying the start and end
 sheet index instead of sheet model
 this.insertSheet([
 index: 1,
 name: 'Inserted Sheet',
 ranges: [{ dataSource:
@Html.Raw(JsonConvert.SerializeObject(@ViewBag.DefaultData)) }],
 columns: [{ width: 150 }, { width: 110 }, { width: 110 }, {
width: 85 }, { width: 85 }, { width: 85 }, { width: 85 },
 { width: 85 }]
]]);
 // Applies style formatting for the inserted sheet
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'Inserted Sheet!A1:H1');
 this.cellFormat({ textAlign: 'center' }, 'Inserted Sheet!D2:H15');
}
</script>

```

### INSERTSHEETCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 }
}

```



```

 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Delete sheet

The Spreadsheet has support for removing an existing worksheet. You can dynamically delete the existing sheet by the following way,

- Right-click on the sheet tab, and then select **Delete** option from context menu.
- Using **delete** method to delete the sheets.

### Rename sheet

You can dynamically rename an existing worksheet in the following way,

- Right-click on the sheet tab, and then select **Rename** option from the context menu.

## Headers

By default, the row and column headers are visible in worksheets. You can dynamically show or hide worksheet headers by using one of the following ways,

- Switch to **View** tab, and then select **Hide Headers** option to hide both the row and column headers.
- Set **showHeaders** property in sheets as **true** or **false** to show or hide the headers at initial load. By default, the **showHeaders** property is enabled in each worksheet.

## Gridlines

Gridlines act as a border like appearance of cells. They are used to distinguish cells on the worksheet. You can dynamically show or hide gridlines by using one of the following ways,

- Switch to **View** tab, and then select **Hide Gridlines** option to hide the gridlines in worksheet.
- Set **showGridLines** property in sheets as **true** or **false** to show or hide the gridlines at initial load. By default, the **showGridLines** property is enabled in each worksheet.

The following code example shows the headers and gridlines operation in spreadsheet.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).Created("created").Sheets(sheet =>
{
 sheet.Name("Shipment
Details").ShowGridLines(false).ShowHeaders(false).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 })
 .Columns(column =>
 {
 column.Width(130).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 })
 .Add();
}).Render()
<script>
function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
 this.cellFormat({ textAlign: 'center' }, 'D2:H15');
 // The gridlines have been removed to set border for the range of
cells
 this.setBorder({ border: '1px solid #e0e0e0' }, 'A1:H15');
}
</script>
```

### HEADERCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynn Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Sheet visibility

Hiding a worksheet can help prevent unauthorized or accidental changes to your file.

There are three visibility state as like Microsoft Excel,

| State | Description |

|-----|-----|

| **Visible** | You can see the worksheet once the component is loaded. |

| **Hidden** | This worksheet is not visible, but you can unhide by selecting the sheet from **List All Sheets** dropdown menu. |

| **VeryHidden** | This worksheet is not visible and cannot be unhidden. Changing the state property to **Visible** is the only way to view this sheet. |

The following code example shows the three types of sheet visibility state.

### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).Created("create
d").Sheets(sheet =>
{
 sheet.Name("Visible Sheet").State(SheetState.Visible).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 sheet.Name("Very Hidden
Sheet").State(SheetState.VeryHidden).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 sheet.Name("Hidden Sheet").State(SheetState.Hidden).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 });
});
```

```

 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
function created() {
 // Applies style formatting to active visible sheet
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:H1');
 this.cellFormat({ textAlign: 'center' }, 'D2:H11');
 // Applies style formatting to active hidden sheet
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' }, 'Hidden
Sheet!A1:H1');
 this.cellFormat({ textAlign: 'center' }, 'Hidden Sheet!D2:H15');
}
</script>

```

### SHEETVISIBILITYCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gerner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 }
}

```

```

 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

See Also

- [Sheet protection](#)
- [Rows and columns](#)
- [Cell range](#)
- [Formatting](#)

## Cell Range in Spreadsheet control

A group of cells in a sheet is known as cell range.

### Wrap text

Wrap text allows you to display large content as multiple lines in a single cell. By default, the wrap text support is enabled. Use the [allowWrap](#) property to enable or disable the wrap text support in spreadsheet.

Wrap text can be applied or removed to a cell or range of cells in the following ways,

- Using the `wrap` property in `cell`, you can enable or disable wrap text to a cell at initial load.
- Select or deselect wrap button from ribbon toolbar to apply or remove the wrap text to the selected range.
- Using the `wrap` method, you can apply or remove the wrap text once the component is loaded.

The following code example shows the wrap text functionality in spreadsheet.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).Created("create
d").Sheets(sheet =>
{
 sheet.Name("Movie List").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Rows(row =>
 {
 row.Height(30).Add();
 row.Cells(cell =>

```

```

 {
 cell.Index(7).Wrap(true).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(7).Wrap(true).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(7).Wrap(true).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(7).Wrap(true).Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(100).Index
 (1).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(150).Add();
 column.Width(120).Add();
 column.Width(90).Add();
 column.Width(180).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:H1');
 this.cellFormat({ verticalAlign: 'middle' }, 'A1:H5');
 this.cellFormat({ textAlign: 'center' }, 'A2:B5');
 this.cellFormat({ textAlign: 'center' }, 'D2:D5');
 // To wrap the cells from E2 to E5 range
 this.wrap('E2:E5');
 // To unwrap the H3 cell
 this.wrap('H3', false);
 }
</script>

```

### WRAPTEXTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { No= "1", ReleasedOn= "1994", Title= "Forrest Gump",
 Rating= "5 Stars", Casts= "Tom Hanks, Robin Wright, Gary Sinise",
 DirectedBy= "Robert Zemeckis", Genre= "Drama", Comments= "Based on the 1986
 novel of the same name by Winston Groom" },
 new { No= "2", ReleasedOn= "1946", Title= "It's a
 Wonderful Life", Rating= "2 Stars", Casts= "James Stewart, Donna Reed,
 Lionel Barrymore", DirectedBy= "Frank Capra", Genre= "Drama", Comments=
 "Colorized version" },
 }
}

```

```

 new { No= "3", ReleasedOn= "1988", Title= "Big", Rating=
"4 Stars", Casts= "Tom Hanks, Elizabeth Perkins, Robert Loggia",
DirectedBy= "Penny Marshall", Genre= "Comedy", Comments= "A thirteen-year-
old boy wishes to be big, and his wish comes true." },
 new { No= "4", ReleasedOn= "1954", Title= "Rear Window",
Rating= "4 Stars", Casts= "James Stewart, Grace Kelly, Wendell Corey",
DirectedBy= "Alfred Hitchcock", Genre= "Suspense", Comments= "Truly
suspenseful and masterfully crafted" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Limitations of Wrap text

The following features have some limitations in wrap text:

- Sorting with wrap text applied data.
- Merge with wrap text

### Merge cells

Merge cells allows users to span two or more cells in the same row or column into a single cell. When cells with multiple values are merged, top-left most cell data will be the data for the merged cell. By default, the merge cells option is enabled. Use [allowMerge](#) property to enable or disable the merge cells option in spreadsheet.

You can merge the range of cells in the following ways,

- Set the `rowSpan` and `colSpan` property in `cell` to merge the number of cells at initial load.
- Select the range of cells and apply merge by selecting the desired option from ribbon toolbar.
- Use `merge` method to merge the range of cells, once the component is loaded.

The available merge options in spreadsheet are,

| Type               | Action                                                           |
|--------------------|------------------------------------------------------------------|
| ----- -----        |                                                                  |
| Merge All          | Combines all the cells in a range in to a single cell (default). |
| Merge Horizontally | Combines cells in a range as row-wise.                           |
| Merge Vertically   | Combines cells in a range as column-wise.                        |
| UnMerge            | Splits the merged cells into multiple cells.                     |

The following code example shows the merge cells operation in spreadsheet.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).Created("create
d").Sheets(sheet =>
{
 sheet.Name("Movie List").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }
}
)

```



```

 }).Rows(row =>
 {
 row.Height(35).Add();
 row.Height(35).Cells(cell =>
 {
 cell.Index(1).RowSpan(2).Add();
 cell.ColSpan(2).Add();
 cell.Index(6).ColSpan(3).Add();
 cell.Index(10).RowSpan(2).ColSpan(3).Add();
 cell.ColSpan(2).Index(13).Add();
 cell.ColSpan(2).Index(17).Add();
 }).Add();
 row.Height(35).Cells(cell =>
 {
 cell.Index(3).ColSpan(3).Add();
 cell.ColSpan(4).Index(6).Add();
 cell.Index(13).ColSpan(3).Add();
 cell.Index(17).ColSpan(2).Add();
 }).Add();
 row.Height(35).Cells(cell =>
 {
 cell.Index(2).ColSpan(3).Add();
 cell.ColSpan(2).Index(5).Add();
 cell.Index(7).ColSpan(3).Add();
 cell.Index(15).ColSpan(2).Add();
 cell.Index(17).ColSpan(2).Add();
 }).Add();
 row.Height(35).Cells(cell =>
 {
 cell.Index(2).ColSpan(3).Add();
 cell.ColSpan(4).Index(6).Add();
 cell.Index(16).ColSpan(2).Add();
 }).Add();
 row.Height(35).Cells(cell =>
 {
 cell.Index(2).ColSpan(4).Add();
 cell.ColSpan(3).Index(7).Add();
 cell.Index(15).ColSpan(2).Add();
 cell.Index(17).ColSpan(2).Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(90).Add();
 column.Width(150).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 }

```

```

 column.Width(120).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:S1');
 this.numberFormat('h:mm AM/PM', 'C1:S1');
 this.cellFormat({ verticalAlign: 'middle' }, 'A1:S11');
 // Merging the `K4:M4` cells using method
 this.merge('K4:M4');
 // Merging the 5th and 6th row cells across 11th, 12th and 13th
column
 this.merge('K5:M6', 'Vertically');
 // Merging the 18th and 19th column cells across 2nd, 3rd and 4th
row
 this.merge('N4:O6', 'Horizontally');
 }
</script>

```

### MERGECELLCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { EmployeeID= "10001", EmployeeName= "Davolio",
 NineAM= "Analysis Task", NinethirtyAM= "Analysis Task", TenAM= "Team
 Meeting", TenthirtyAM= "Testing", ElevenAM= "Development", EleventhirtyAM=
 "Development", TwelvePM= "Development", TwelvethirtyPM= "Support", OnePM=
 "Lunch Break", OnethirtyPM= "Lunch Break", TwoPM= "Lunch Break",
 TwothirtyPM= "Testing", ThreePM= "Testing", ThreethirtyPM= "Development",
 FourPM= "Conference", FourthirtyPM= "Team Meeting", FivePM= "Team Meeting"
 },
 new { EmployeeID= "10002", EmployeeName= "Buchanan",
 NineAM= "Task Assign", NinethirtyAM= "Support", TenAM= "Support",
 TenthirtyAM= "Support", ElevenAM= "Testing", EleventhirtyAM= "Testing",
 TwelvePM= "Testing", TwelvethirtyPM= "Testing", OnePM= "Lunch Break",
 OnethirtyPM= "Lunch Break", TwoPM= "Lunch Break", TwothirtyPM=
 "Development", ThreePM= "Development", ThreethirtyPM= "Check Mail",
 FourPM= "Check Mail", FourthirtyPM= "Team Meeting", FivePM= "Team Meeting"
 },
 new { EmployeeID= "10003", EmployeeName= "Fuller", NineAM=
 "Check Mail", NinethirtyAM= "Check Mail", TenAM= "Check Mail",
 TenthirtyAM= "Analysis Tasks", ElevenAM= "Analysis Tasks", EleventhirtyAM=
 "Support", TwelvePM= "Support", TwelvethirtyPM= "Support", OnePM= "Lunch
 Break", OnethirtyPM= "Lunch Break", TwoPM= "Lunch Break", TwothirtyPM=
 "Development", ThreePM= "Development", ThreethirtyPM= "Team Meeting",
 FourPM= "Team Meeting", FourthirtyPM= "Development", FivePM= "Development"
 },
 new { EmployeeID= "10004", EmployeeName= "Leverling",
 NineAM= "Testing", NinethirtyAM= "Check Mail", TenAM= "Check Mail",

```

```

TenthirtyAM= "Support", ElevenAM= "Testing", EleventhirtyAM= "Testing",
TwelvePM= "Testing", TwelvethirtyPM= "Testing", OnePM= "Lunch Break",
OnethirtyPM= "Lunch Break", TwoPM= "Lunch Break", TwothirtyPM=
"Development", ThreePM= "Development", ThreethirtyPM= "Check Mail",
FourPM= "Conference", FourthirtyPM= "Conference", FivePM= "Team Meeting" },
 new { EmployeeID= "10005", EmployeeName= "Peacock",
NineAM= "Task Assign", NinethirtyAM= "Task Assign", TenAM= "Task Assign",
TenthirtyAM= "Task Assign", ElevenAM= "Check Mail", EleventhirtyAM=
"Support", TwelvePM= "Support", TwelvethirtyPM= "Support", OnePM= "Lunch
Break", OnethirtyPM= "Lunch Break", TwoPM= "Lunch Break", TwothirtyPM=
"Development", ThreePM= "Development", ThreethirtyPM= "Team Meeting",
FourPM= "Team Meeting", FourthirtyPM= "Testing", FivePM= "Testing" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Limitations of Merge

The following features have some limitations in Merge:

- Merge with filter.
- Merge with wrap text.

### Data Validation

Data Validation is used to restrict the user from entering the invalid data. You can use the [allowDataValidation](#) property to enable or disable data validation.

**Note:** \* The default value for `allowDataValidation` property is `true`.

### Apply Validation

You can apply data validation to restrict the type of data or the values that users enter into a cell.

You can apply data validation by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Data Validation item.
- Use the `addDataValidation()` method programmatically.

### Clear Validation

Clear validation feature is used to remove data validations from the specified ranges or the whole worksheet.

You can clear data validation rule by one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Validation item.
- Use the `removeDataValidation()` method programmatically.

### Highlight Invalid Data

Highlight invalid data feature is used to highlight the previously entered invalid values.

You can highlight an invalid data by using one of the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Highlight Invalid Data item.

- Use the `addInvalidHighlight()` method programmatically.

#### *Clear Highlighted Invalid Data*

Clear highlight feature is used to remove the highlight from invalid cells.

You can clear the highlighted invalid data by using the following ways,

- Select the Data tab in the Ribbon toolbar, and then choose the Clear Highlight item.
- Use the `removeInvalidHighlight()` method programmatically.

#### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).Created("created").Sheets(sheet =>
{
 sheet.Name("PriceDetails").Rows(row =>
 {
 row.Cells(cell =>
 {
 cell.Value("Seller Name").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Customer Id").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Customer Name").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Product Name").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Product Price").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Sales Date").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Billing Time").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Total Price").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("John").Add();
 cell.Value("1").Validation(new SpreadsheetValidation() { Type =
 ValidationType.WholeNumber, Operator = ValidationOperator.NotEqualTo, Value1
 = "1" }).Add();
 cell.Value("Nash").Add();
 cell.Value("Digger").Validation(new SpreadsheetValidation() {
 Type = ValidationType.List, Value1 = "Digger, Digger, Cherrypicker" }
).Add();
 cell.Value("50000").Validation(new SpreadsheetValidation() {
 Type = ValidationType.List, Value1 = "50000,50000,45000" }).Add();
 cell.Value("04/11/2019").Add();
 cell.Value("11:34:32 AM").Add();
 cell.Value("1,45,000.00").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Mike").Add();
```

```

 cell.Value("2").Validation(new SpreadsheetValidation() { Type =
ValidationType.WholeNumber, Operator = ValidationOperator.NotEqualTo, Value1
= "1" }).Add();
 cell.Value("Jim").Add();
 cell.Value("Cherrypicker").Validation(new
SpreadsheetValidation() { Type = ValidationType.List, Value1 =
"Cherrypicker, JCB, Wheelbarrow" }).Add();
 cell.Value("45000").Validation(new SpreadsheetValidation() {
Type = ValidationType.List, Value1 = "45000,90000,40" }).Add();
 cell.Value("04/11/2019").Add();
 cell.Value("11:34:32 AM").Add();
 cell.Value("1,45,000.00").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("shane").Add();
 cell.Value("3").Validation(new SpreadsheetValidation() { Type =
ValidationType.WholeNumber, Operator = ValidationOperator.NotEqualTo, Value1
= "1" }).Add();
 cell.Value("Sean").Add();
 cell.Value("Kango").Validation(new SpreadsheetValidation() {
Type = ValidationType.List, Value1 = "Kango, Ropes" }).Add();
 cell.Value("450").Validation(new SpreadsheetValidation() { Type
= ValidationType.List, Value1 = "450, 95" }).Add();
 cell.Value("06/25/2019").Add();
 cell.Value("01:30:11 PM").Add();
 cell.Value("545.00").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("John").Add();
 cell.Value("1").Validation(new SpreadsheetValidation() { Type =
ValidationType.WholeNumber, Operator = ValidationOperator.NotEqualTo, Value1
= "1" }).Add();
 cell.Value("Nash").Add();
 cell.Value("JCB").Validation(new SpreadsheetValidation() { Type
= ValidationType.List, Value1 = "JCB, Ropes, scaffolding" }).Add();
 cell.Value("90000").Validation(new SpreadsheetValidation() {
Type = ValidationType.List, Value1 = "90000, 95, 10000" }).Add();
 cell.Value("09/22/2019").Add();
 cell.Value("12:30:02 PM").Add();
 cell.Value("1,00,095.00").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(88).Add();
 column.Width(88).Add();
 column.Width(106).Add();
 column.Width(98).Add();
 column.Width(88).Add();
 column.Width(86).Add();
 column.Width(107).Add();
 column.Width(81).Add();
 }).Add();
}).Render()
<script>
function created() {

```

```

 //Add Data Validation to range.
 this.addDataValidation({ type: 'TextLength', Operator:
'LessThanOrEqualTo', Value1: '4' }, 'A2:A5');
 this.addDataValidation({ type: 'WholeNumber', Operator:
'NotEqualTo', Value1: '1' }, 'B2:B5');
 this.addDataValidation({ type: 'Date', Operator: 'NotEqualTo',
Value1: '04/11/2019' }, 'F2:F5');
 this.addDataValidation({ type: 'Time', Operator: 'Between', Value1:
'10:00:00 AM', value2: '11:00:00 AM' }, 'G2:G5');
 this.addDataValidation({ type: 'Decimal', Operator: 'LessThan',
Value1: '100000.00' }, 'H2:H5');
 //Highlight Invalid Data.
 this.addInvalidHighlight('A1:H5');
 }
</script>

```

### DATAVALIDATION.CS

```

public IActionResult Index()
{
 return View();
}

```

#### Limitations of Data validation

The following features have some limitations in Data Validation:

- Entire row data validation.
- Insert row between the data validation.
- Copy/paste with data validation.
- Delete cells between data validation applied range.

#### Auto Fill

Auto Fill is used to fill the cells with data based on adjacent cells. It also follows a pattern from adjacent cells if available. There is no need to enter the repeated data manually. You can use [allowAutoFill](#) property to enable/disable the auto fill support. You can also use [showFillOptions](#) property to enable/disable the fill option and [fillType](#) property to change the default auto fill option which is available in [autoFillSettings](#).

You can do this by one of the following ways,

- Using “AutoFillOptions” menu which is open, while drag and drop the cell using fill handle element.
- Use the `autoFill()` method programmatically.

The available parameters in `autoFill()` method are,

| Parameter | Type   | Description               |
|-----------|--------|---------------------------|
| dataRange | string | Specifies the data range. |

| fillRange | string | Specifies the fill range. |

| direction | AutoFillDirection | Specifies the direction("Up","Right","Down","Left")to be filled. |

| fillType | AutoFillType | Specifies the fill type("CopyCells","FillSeries","FillFormattingOnly","FillWithoutFormatting") for autofill action. |

In Auto Fill we have following options,

- Copy Cells
- Fill Series
- Fill Formatting Only
- Fill Without Formatting

**Note:** \* The default auto fill option is “FillSeries” which can be referred from fillType property.

#### *Copy Cells*

To copy the selected cell content to the adjacent cells. You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Copy Cells” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “CopyCells” as fill type in autoFill method to fill the adjacent cells.

#### *Fill Series*

To fill the series of numbers, characters, or dates based on selected cell content to the adjacent cells with their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Series” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillSeries” as fill type in autoFill method to fill the adjacent cells.

#### *Fill Formatting Only*

To fill the cell style and number formatting based on the selected cell content to the adjacent cells without their content.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Formatting Only” option in “AutoFillOptions” menu to fill the adjacent cells.
- Using “FillFormattingOnly” as fill type in autoFill method to fill the adjacent cells.

#### *Fill Without Formatting*

To fill series of numbers, characters, or dates based on the selected cells to the adjacent cells without their formats.

You can do this by one of the following ways,

- Using fill handle to select the adjacent cell range and “Fill Without Formatting” option in “AutoFillOptions” menu to fill the adjacent cells.

- Using “FillWithoutFormatting” as fill type in `autoFill` method to fill the adjacent cells.

In the following sample, you can enable/disable the fill option on the button click event by using the `showFillOptions` property in `autoFillSettings`.

### CSHTML

```
@Html.EJS().Button("showfillbtn").Content("Change
showFillOptions").Render();
@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
{
 sheet.Name("Price Details").Rows(row =>
 {
 row.Height(30).Add();
 }).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A1").
 Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ backgroundColor: '#357cd2', color: '#fff',
fontWeight: 'bold', textAlign: 'center' }, 'A1:H1');
 this.autoFill('D2:D3', 'D4:D11', 'Down', 'CopyCells');
 this.autoFill('E2:E3', 'E4:E11', 'Down', 'FillSeries');
 this.autoFill('B2:B3', 'B4:B11', 'Down', 'FillFormattingOnly');
 this.autoFill('C2:C3', 'C4:C11', 'Down', 'FillWithoutFormatting');
 }
 document.getElementById("showfillbtn").addEventListener('click',
function () {
 var spreadsheetObj =
document.getElementById("spreadsheet").ej2_instances[0];
 var showFillOptions =
spreadsheetObj.autoFillSettings.showFillOptions;
 spreadsheetObj.autoFillSettings.showFillOptions =
!showFillOptions;
 });
</script>
```

### AUTOFILLCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> defaultData = new List<object>()
 {
 new { Item Name= "Casual Shoes", Date= "02/14/2014", Time=
"11=34=32 AM", Quantity= "10", Price= "20", Amount= "200", Discount= "1",
Profit= "10" },
 }
```



```

 new { Item Name= "Sports Shoes", Date= "06/11/2014", Time=
"05=56=32 AM", Quantity= "20", Price= "30", Amount= "600", Discount= "5",
Profit= "50" },
 new { Item Name= "Formal Shoes", Date= "07/27/2014", Time=
"03=32=44 AM", Quantity= "20", Price= "15", Amount= "300", Discount= "7",
Profit= "27" },
 new { Item Name= "Sandals & Floaters", Date= "11/21/2014",
Time= "06=23=54 AM", Quantity= "15", Price= "20", Amount= "300", Discount=
"11", Profit= "67" },
 new { Item Name= "Flip- Flops & Slippers", Date=
"06/23/2014", Time= "12=43=59 AM", Quantity= "30", Price= "10", Amount=
"300", Discount= "10", Profit= "70" },
 new { Item Name= "Sneakers", Date= "07/22/2014", Time=
"10=55=53 AM", Quantity= "40", Price= "20", Amount= "800", Discount= "13",
Profit= "66" },
 new { Item Name= "Running Shoes", Date= "02/04/2014", Time=
"03=44=34 AM", Quantity= "20", Price= "10", Amount= "200", Discount= "3",
Profit= "14" },
 new { Item Name= "Loafers", Date= "11/30/2014", Time=
"03=12=52 AM", Quantity= "31", Price= "10", Amount= "310", Discount= "6",
Profit= "29" },
 new { Item Name= "Cricket Shoes", Date= "07/09/2014", Time=
"11=32=14 AM", Quantity= "41", Price= "30", Amount= "1210", Discount= "12",
Profit= "166" },
 new { Item Name= "T-Shirts", Date= "10/31/2014", Time=
"12=01=44 AM", Quantity= "50", Price= "10", Amount= "500", Discount= "9",
Profit= "55" }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Clear

Clear feature helps you to clear the cell contents (formulas and data), formats (including number formats, conditional formats, and borders) in a spreadsheet. When you apply clear all, both the contents and the formats will be cleared simultaneously.

### Apply Clear Feature

You can apply clear feature by using one of the following ways,

- Select the clear icon in the Ribbon toolbar under the Home Tab.
- Using the `clear()` method to clear the values.

Clear has the following types in the spreadsheet,

| Options | Uses |

|-----|-----|

| **Clear All** | Used to clear all contents, formats, and hyperlinks. |

| **Clear Formats** | Used to clear the formats (including number formats, conditional formats, and borders) in a cell. |

| **Clear Contents** | Used to clear the contents (formulas and data) in a cell. |

| **Clear Hyperlinks** | Used to clear the hyperlink in a cell. |

### Methods

Clear the cell contents and formats in the Spreadsheet document by using the **clear** method. The clear method has **type** and **range** as parameters. The following code example shows how to clear the cell contents and formats in the button click event.

### CSHTML

```
@Html.EJS().DropDownButton("element").Content("Clear").Items((IEnumerable<object>)ViewBag.items).Select("itemSelect").Render()
@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
function created() {
 this.cellFormat({ fontWeight: 'bold', fontSize: '12pt', 'A1:E1' });
 this.cellFormat({ color: '#10c469' }, 'B1:B10');
}

function itemSelect(args) {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (args.item.text === 'Clear All')
 spreadsheet.clear({ type: 'Clear All', range: 'D1:D10' }); // Clear
the content, formats and hyperlinks applied in the provided range.
 if (args.item.text === 'Clear Formats')
 spreadsheet.clear({ type: 'Clear Formats', range: 'B1:B10' }); //
Clear the formats applied in the provided range
 if (args.item.text === 'Clear Contents')
 spreadsheet.clear({ type: 'Clear Contents', range: 'A1:A10' }); //
Clear the content in the provided range
 if (args.item.text === 'Clear Hyperlinks')
 spreadsheet.clear({ type: 'Clear Hyperlinks', range: 'F2:F6' }); //
Clear the hyperlinks applied in the provided range
}
</script>
```

### CLEARCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { OrderID= "10248", CustomerID= "VINET", EmployeeID=
"5", ShipName= "Vins et alcools Chevalier", ShipCity= "Reims", Website=
"https://www.amazon.com/" },
 new { OrderID= "10249", CustomerID= "TOMSP", EmployeeID=
"6", ShipName= "Toms Spezialitäten", ShipCity= "Münster", Website=
"https://www.overstock.com/" },
 new { OrderID= "10250", CustomerID= "HANAR", EmployeeID=
"4", ShipName= "Hanari Carnes", ShipCity= "Rio de Janeiro", Website=
"https://www.aliexpress.com/" },
 }
```

```

 new { OrderID= "10251", CustomerID= "VICTE", EmployeeID=
"3", ShipName= "Victuailles en stock", ShipCity= "Lyon", Website=
"http://www.alibaba.com/" },
 new { OrderID= "10252", CustomerID= "SUPRD", EmployeeID=
"4", ShipName= "Suprêmes délices", ShipCity= "Charleroi", Website=
"https://taobao.com/" },

 };
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Clear All"
 });
 items.Add(new
 {
 text = "Clear Formats"
 });
 items.Add(new
 {
 text = "Clear Contents"
 });
 items.Add(new
 {
 text = "Clear Hyperlinks"
 });
 ViewBag.items = items;
 ViewBag.DefaultData = data;
 return View();
}

```

## See Also

- [Rows and columns](#)
- [Formatting](#)
- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)

## Editing in Spreadsheet control

You can edit the contents of a cell directly in the cell or by typing in the formula bar. By default, the editing feature is enabled in the spreadsheet. Use the [allowEditing](#) property to enable or disable the editing feature.

### Edit cell

You can start editing by one of the following ways,

- Double click a cell to start the edit mode.
- Press **F2** key to edit the active cell.
- Use formula bar to perform editing.
- Use **BACKSPACE** or **SPACE** key to clear the cell content and start the edit mode.
- Using the **startEdit** method.

### Save cell

If the cell is in editable state, you can save the edited cell by one of the following ways,

- Perform mouse click on any other cell rather than the current editing cell.
- Press **Enter** or **Tab** keys to save the edited cell content.
- Using the `endEdit` method.

### Cancel editing

To cancel the editing without saving the changes, you can use one of the following ways,

- Press **ESCAPE** key, this will remove the editable state and update the unchanged cell content.
- Using the `closeEdit` method.

The following sample shows how to prevent the editing and cell save. Here **E** column prevent the editing by using cancel argument as true in `cellEdit` event. In **D** column, prevent saving the edited changes by using cancel argument as true in `beforeCellSave` and use `closeEdit` method in spreadsheet.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.CellEdit("cellEdit").BeforeCellSave("beforeCellSave").Sheets(sheet =>
{
 sheet.SelectedRange("C7").Name("Movie List").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Rows(row =>
 {
 row.Index(10).Cells(cell =>
 {
 cell.Index(3).Value("Total Amount:").Style(new
SpreadsheetCellStyle() { FontWeight = FontWeight.Bold }).Add();
 cell.Formula("=SUM(E2:E10)").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(120).Add();
 column.Width(180).Add();
 column.Width(100).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:E1');
 this.cellFormat({ textAlign: 'center' }, 'A2:A10');
 this.cellFormat({ textAlign: 'center' }, 'C2:C10');
 this.numberFormat('$#,##0.00', 'D2:D10');
 this.numberFormat('$#,##0.00', 'E2:E11');
 }
 function cellEdit(args) {
 // Preventing the editing in 5th(Amount) column.
```

```

 if (args.address.includes('E')) { args.cancel = true; }
 }
 function beforeCellSave(args) {
 // Prevent saving the edited changes in 4th(Rate) column.
 if (args.address.includes('D')) {
 args.cancel = true;
 // Manually removes the editable state without saving the
 changes. Use `endEdit` method if you want to save the changes.
 this.closeEdit();
 }
 }
}
</script>

```

## EDITINGCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ItemCode= "I231", ItemName= "Chinese Combo Noodle",
Quantity= "2", Rate= "125", Amount= "=PRODUCT(C2,D2)" },
 new { ItemCode= "I245", ItemName= "Chinese Combo Rice",
Quantity= "3", Rate= "125", Amount= "=PRODUCT(C3,D3)" },
 new { ItemCode= "I237", ItemName= "Amritsari Chola",
Quantity= "2", Rate= "225", Amount= "=PRODUCT(C4,D4)" },
 new { ItemCode= "I291", ItemName= "Asian Mixed Entree
Platt", Quantity= "3", Rate= "165", Amount= "=PRODUCT(C5,D5)" },
 new { ItemCode= "I268", ItemName= "Chinese Combo Chicken",
Quantity= "3", Rate= "125", Amount= "=PRODUCT(C6,D6)" },
 new { ItemCode= "I251", ItemName= "Chivas Regal",
Quantity= "1", Rate= "325", Amount= "=PRODUCT(C7,D7)" },
 new { ItemCode= "I256", ItemName= "Chicken Drumsticks",
Quantity= "2", Rate= "180", Amount= "=PRODUCT(C8,D8)" },
 new { ItemCode= "I232", ItemName= "Manchow Soup",
Quantity= "2", Rate= "160", Amount= "=PRODUCT(C9,D9)" },
 new { ItemCode= "I290", ItemName= "Schezuan Chicken",
Quantity= "3", Rate= "180", Amount= "=PRODUCT(C10,D10)" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Limitations

- Text overflow in cells is not supported in Editing.

## See Also

- [Cell range](#)
- [Formatting](#)
- [Hyperlink](#)
- [Undo and Redo](#)
- [Unlock the particular cells in the protected sheet](#)

## Formulas in Spreadsheet control

Formulas are used for calculating the data in a worksheet. You can refer the cell reference from same sheet or from different sheets.

### Usage

You can set formula for a cell in the following ways,

- Using the `formula` property from `cell`, you can set the formula or expression to each cell at initial load.
- Set the formula or expression through data binding.
- You can set formula for a cell by [editing](#).
- Using the `updateCell` method, you can set or update the cell formula.

### Create User Defined Functions / Custom Functions

The Spreadsheet includes a number of built-in formulas. For your convenience, a list of supported formulas can be found [here](#).

You can define and use an unsupported formula, i.e. a user defined/custom formula, in the spreadsheet by using the `addCustomFunction` function. Meanwhile, remember that you should define a user defined/custom formula whose results should only return a single value. If a user-defined/custom formula returns an array, it will be time-consuming to update adjacent cell values.

The following code example shows an unsupported formula in the spreadsheet.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).StartCell("A2").
 Add();
 }).Rows(row =>
 {
 row.Height(40).CustomHeight(true).Cells(cell =>
 {
 cell.ColSpan(5).Value("Monthly Expense").Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, VerticalAlign =
 VerticalAlign.Middle, TextAlign=TextAlign.Center, FontSize="15pt",
 FontStyle= FontStyle.Italic }).Add();
 }).Add();
 row.Height(30).Add();
 row.Index(11).Cells(cell =>
 {
 cell.Value("Totals").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, FontStyle= FontStyle.Italic }).Add();
 cell.Formula("=SUM(B3:B11)").Add();
 cell.Formula("=SUM(C3:C11)").Add();
 cell.Formula("=SUM(D3:D11)").Add();
 }).Add();
 row.Cells(cell =>
 {
```

```

 cell.Index(1).Value("Number of Categories").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=COUNTA(A3:A11)").Index(3).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Average Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=AVERAGE(B3:B11)").Index(3).Format("$#,##0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Min Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=MIN(B3:B11)").Index(3).Format("$#,##0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Max Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=MAX(B3:B11)").Index(3).Format("$#,##0").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(150).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(140).Add();
 column.Width(150).Add();
 }).Add();
 }).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A2:F2');
 this.numberFormat('$#,##0', 'B3:D12');
 this.numberFormat('0%', 'E3:E12');
 // Adding custom function for calculating the percentage between two
 cells.
 this.addCustomFunction(calculatePercentage, 'PERCENTAGE');
 // Adding custom function for calculating round down for the value.
 this.addCustomFunction(roundDownHandler, 'ROUNDDOWN');
 // Calculate percentage using custom added formula in E12 cell.
 this.updateCell({ formula: '=PERCENTAGE(C12,D12)' }, 'E12');
 // Calculate round down for average values using custom added
 formula in F12 cell.
 this.updateCell({ formula: '=ROUNDDOWN(F11,1)' }, 'F12');
 }
 // Custom function to calculate percentage between two cell values.
 function calculatePercentage(firstCell,secondCell) {
 return (firstCell) / (secondCell);
 }

```

```

 }
 // Custom function to calculate round down for values.
 function roundDownHandler(value, digit) {
 var multiplier = Math.pow(10, digit);
 return Math.floor(value * multiplier) / multiplier;
 }
</script>

```

### FORMULACONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Category= "Household Utilities", MonthlySpend=
 "=C3/12", AnnualSpend= "3000", LastYearSpend= "3000", PercentageChange=
 "=C3/D3", AverageChange= "=7.9/E3"},
 new { Category= "Food", MonthlySpend= "=C4/12",
 AnnualSpend= "2500", LastYearSpend= "2250", PercentageChange= "=C4/D4",
 AverageChange= "=7.9/E4"},
 new { Category= "Gasoline", MonthlySpend= "=C5/12",
 AnnualSpend= "1500", LastYearSpend= "1200", PercentageChange= "=C5/D5",
 AverageChange= "=7.9/E5"},
 new { Category= "Clothes", MonthlySpend= "=C6/12",
 AnnualSpend= "1200", LastYearSpend= "1000", PercentageChange= "=C6/D6",
 AverageChange= "=7.9/E6"},
 new { Category= "Insurance", MonthlySpend= "=C7/12",
 AnnualSpend= "1500", LastYearSpend= "1500", PercentageChange= "=C7/D7",
 AverageChange= "=7.9/E7"},
 new { Category= "Taxes", MonthlySpend= "=C8/12",
 AnnualSpend= "3500", LastYearSpend= "3500", PercentageChange= "=C8/D8",
 AverageChange= "=7.9/E8"},
 new { Category= "Entertainment", MonthlySpend= "=C9/12",
 AnnualSpend= "2000", LastYearSpend= "2250", PercentageChange= "=C9/D9",
 AverageChange= "=7.9/E9"},
 new { Category= "Vacation", MonthlySpend= "=C10/12",
 AnnualSpend= "1500", LastYearSpend= "2000", PercentageChange= "=C10/D10",
 AverageChange= "=7.9/E10"},
 new { Category= "Miscellaneous", MonthlySpend= "=C11/12",
 AnnualSpend= "1250", LastYearSpend= "1558", PercentageChange= "=C11/D11",
 AverageChange= "=7.9/E11"},
 };
 ViewBag.DefaultData = data;
 return View();
}

```

Second, if you want to directly compute any formula or expression, you can use the [computeExpression](#) method. This method will work for both built-in and used-defined/custom formula.

The following code example shows how to use `computeExpression` method in the spreadsheet.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).Sheets(sheet =>
{

```



```

sheet.Ranges(ranges =>
{
ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A2").
Add();
}).Rows(row =>
{
 row.Height(40).CustomHeight(true).Cells(cell =>
 {
 cell.ColSpan(5).Value("Monthly Expense").Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, VerticalAlign =
 VerticalAlign.Middle, TextAlign=TextAlign.Center, FontSize="15pt",
 FontStyle= FontStyle.Italic }).Add();
 }).Add();
 row.Height(30).Add();
 row.Index(11).Cells(cell =>
 {
 cell.Value("Totals").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, FontStyle= FontStyle.Italic }).Add();
 cell.Formula("=SUM(B3:B11)").Add();
 cell.Formula("=SUM(C3:C11)").Add();
 cell.Formula("=SUM(D3:D11)").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Number of Categories").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=COUNTA(A3:A11)").Index(3).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Average Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=AVERAGE(B3:B11)").Index(3).Format("$#,##0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Min Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=MIN(B3:B11)").Index(3).Format("$#,##0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Max Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign=
 TextAlign.Right }).Add();
 cell.Formula("=MAX(B3:B11)").Index(3).Format("$#,##0").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(150).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 }
}

```

```

 column.Width(120).Add();
 column.Width(140).Add();
 column.Width(150).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A2:E2');
 this.numberFormat('$#,##0', 'B3:D12');
 this.numberFormat('0%', 'E3:E12');
 // Adding custom function for calculating the percentage between two
 cells.
 this.addCustomFunction(calculatePercentage, 'PERCENTAGE');
 // Calculate percentage using custom added formula in E11 cell.
 this.updateCell({ formula: '=PERCENTAGE(C11,D11)' }, 'E11');
 // Calculate expressions using computeExpression in E10 cell.
 this.updateCell({ value: this.computeExpression('C10/D10') }, 'E10');
 // Calculate custom formula values using computeExpression in E12
 cell.
 this.updateCell({ value:
 this.computeExpression('=PERCENTAGE(C12,D12)'), }, 'E12');
 // Calculate SUM (built-in) formula values using computeExpression
 in D12 cell.
 this.updateCell({ value: this.computeExpression('=SUM(D3:D11)') },
 'D12');
 }
 // Custom function to calculate percentage between two cell values.
 function calculatePercentage(firstCell,secondCell) {
 return (firstCell) / (secondCell);
 }
</script>

```

### FORMULACONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Category= "Household Utilities", MonthlySpend=
 "=C3/12", AnnualSpend= "3000", LastYearSpend= "3000", PercentageChange=
 "=C3/D3"},
 new { Category= "Food", MonthlySpend= "=C4/12",
 AnnualSpend= "2500", LastYearSpend= "2250", PercentageChange= "=C4/D4"},
 new { Category= "Gasoline", MonthlySpend= "=C5/12",
 AnnualSpend= "1500", LastYearSpend= "1200", PercentageChange= "=C5/D5"},
 new { Category= "Clothes", MonthlySpend= "=C6/12",
 AnnualSpend= "1200", LastYearSpend= "1000", PercentageChange= "=C6/D6"},
 new { Category= "Insurance", MonthlySpend= "=C7/12",
 AnnualSpend= "1500", LastYearSpend= "1500", PercentageChange= "=C7/D7"},
 new { Category= "Taxes", MonthlySpend= "=C8/12",
 AnnualSpend= "3500", LastYearSpend= "3500", PercentageChange= "=C8/D8"},
 new { Category= "Entertainment", MonthlySpend= "=C9/12",
 AnnualSpend= "2000", LastYearSpend= "2250", PercentageChange= "=C9/D9"},
 new { Category= "Vacation", MonthlySpend= "=C10/12",
 AnnualSpend= "1500", LastYearSpend= "2000", PercentageChange= "=C10/D10"},
 }
}

```

```

 new { Category= "Miscellaneous", MonthlySpend= "=C11/12",
AnnualSpend= "1250", LastYearSpend= "1558", PercentageChange= "=C11/D11"},
 };
 ViewBag.DefaultData = data;
 return View();
 }

```

### Formula bar

Formula bar is used to edit or enter cell data in much easier way. By default, the formula bar is enabled in the spreadsheet. Use the [showFormulaBar](#) property to enable or disable the formula bar.

### Named Ranges

You can define a meaningful name for a cell range and use it in the formula for calculation. It makes your formula much easier to understand and maintain. You can add named ranges to the Spreadsheet in the following ways,

- Using the [definedNames](#) collection, you can add multiple named ranges at initial load.
- Use the `addDefinedName` method to add a named range dynamically.
- You can remove an added named range dynamically using the `removeDefinedName` method.
- Select the range of cells, and then enter the name for the selected range in the name box.

The following code example shows the usage of named ranges support.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).BeforeDataBound(
"beforeDataBound").Created("created").ShowRibbon(false).Sheets(sheet =>
{
 sheet.Name("Budget Details").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A2").
Add();
 }).Rows(row =>
 {
 row.Height(40).CustomHeight(true).Cells(cell =>
 {
 cell.ColSpan(5).Value("Monthly Expense").Style(new
SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, VerticalAlign =
VerticalAlign.Middle, TextAlign = TextAlign.Center, FontSize = "15pt",
FontStyle = FontStyle.Italic }).Add();
 }).Add();
 row.Height(30).Add();
 row.Index(11).Cells(cell =>
 {
 cell.Value("Totals").Style(new SpreadsheetCellStyle() {
FontWeight = FontWeight.Bold, FontStyle = FontStyle.Italic }).Add();
 cell.Formula("=SUM(MonthlySpending)") .Add();
 cell.Formula("=SUM(AnnualSpending)") .Add();
 cell.Formula("=SUM(LastYearSpending)") .Add();
 cell.Formula("=C12/D12") .Add();
 }).Add();
 row.Cells(cell =>
 {

```

```

 cell.Index(1).Value("Number of Categories").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign =
 TextAlign.Right }).Add();
 cell.Formula("=COUNTA(=C12/D12)").Index(3).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Average Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign =
 TextAlign.Right }).Add();

cell.Formula("=AVERAGE(MonthlySpending)").Index(3).Format("$#,##0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Min Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign =
 TextAlign.Right }).Add();

cell.Formula("=MIN(MonthlySpending)").Index(3).Format("$#,##0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(1).Value("Max Spend").ColSpan(2).Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, TextAlign =
 TextAlign.Right }).Add();

cell.Formula("=MAX(MonthlySpending)").Index(3).Format("$#,##0").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(150).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 column.Width(120).Add();
 }).Add();
}).DefinedNames(definedName => {
 definedName.Name("Categories").RefersTo("=Budget
Details!A3:A11").Add();
 definedName.Name("MonthlySpending").RefersTo("=Budget
Details!B3:B11").Add();
 definedName.Name("AnnualSpending").RefersTo("=Budget
Details!C3:C11").Add();
}).Render()
<script>
function created() {
 // Removing the unwanted `PercentageChange` named range
 this.removeDefinedName('PercentageChange', '');
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A2:E2');
 this.numberFormat('$#,##0', 'B3:D12');
 this.numberFormat('0%', 'E3:E12');
}
function beforeDataBound() {
 // Adding name dynamically for `last year spending` and `percentage
 change` ranges.

```

```

 this.addDefinedName({ name: 'LastYearSpending', refersTo: '=D3:D11'
 });
 this.addDefinedName({ name: 'PercentageChange', refersTo: '=E3:E11'
 });
 }
</script>

```

### DEFINEDNAMECONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Category= "Household Utilities", MonthlySpend=
 "=C3/12", AnnualSpend= "3000", LastYearSpend= "3000", PercentageChange=
 "=C3/D3"},
 new { Category= "Food", MonthlySpend= "=C4/12",
 AnnualSpend= "2500", LastYearSpend= "2250", PercentageChange= "=C4/D4"},
 new { Category= "Gasoline", MonthlySpend= "=C5/12",
 AnnualSpend= "1500", LastYearSpend= "1200", PercentageChange= "=C5/D5"},
 new { Category= "Clothes", MonthlySpend= "=C6/12",
 AnnualSpend= "1200", LastYearSpend= "1000", PercentageChange= "=C6/D6"},
 new { Category= "Insurance", MonthlySpend= "=C7/12",
 AnnualSpend= "1500", LastYearSpend= "1500", PercentageChange= "=C7/D7"},
 new { Category= "Taxes", MonthlySpend= "=C8/12",
 AnnualSpend= "3500", LastYearSpend= "3500", PercentageChange= "=C8/D8"},
 new { Category= "Entertainment", MonthlySpend= "=C9/12",
 AnnualSpend= "2000", LastYearSpend= "2250", PercentageChange= "=C9/D9"},
 new { Category= "Vacation", MonthlySpend= "=C10/12",
 AnnualSpend= "1500", LastYearSpend= "2000", PercentageChange= "=C10/D10"},
 new { Category= "Miscellaneous", MonthlySpend= "=C11/12",
 AnnualSpend= "1250", LastYearSpend= "1558", PercentageChange= "=C11/D11"},
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Supported Formulas

The following are the list of formulas supported in spreadsheet,

| Formula | Description |
|---------|-------------|
|---------|-------------|

|       |       |
|-------|-------|
| ----- | ----- |
|-------|-------|

|     |                                                 |
|-----|-------------------------------------------------|
| ABS | Returns the value of a number without its sign. |
|-----|-------------------------------------------------|

|         |                                                                           |
|---------|---------------------------------------------------------------------------|
| ADDRESS | Returns a cell reference as text, given specified row and column numbers. |
|---------|---------------------------------------------------------------------------|

|     |                                                                      |
|-----|----------------------------------------------------------------------|
| AND | Returns TRUE if all the arguments are TRUE, otherwise returns FALSE. |
|-----|----------------------------------------------------------------------|

|         |                                                                           |
|---------|---------------------------------------------------------------------------|
| AVERAGE | Calculates average for the series of numbers and/or cells excluding text. |
|---------|---------------------------------------------------------------------------|

|          |                                                                                 |
|----------|---------------------------------------------------------------------------------|
| AVERAGEA | Calculates the average for the cells evaluating TRUE as 1, text and FALSE as 0. |
|----------|---------------------------------------------------------------------------------|

|           |                                                          |
|-----------|----------------------------------------------------------|
| AVERAGEIF | Clears content of the active cell and enables edit mode. |
|-----------|----------------------------------------------------------|

|            |                                                                 |
|------------|-----------------------------------------------------------------|
| AVERAGEIFS | Calculates average for the cells based on specified conditions. |
|------------|-----------------------------------------------------------------|

| CEILING | Rounds a number up to the nearest multiple of a given factor. |

| CHOOSE | Returns a value from list of values, based on index number. |

| CHAR | Returns the character from the specified number. |

| CODE | Returns the numeric code for the first character in a given string. |

| CONCAT | Concatenates a list or a range of text strings. |

| CONCATENATE | Combines two or more strings together. |

| COUNT | Counts the cells that contain numeric values in a range. |

| COUNTA | Counts the cells that contains values in a range. |

| COUNTBLANK | Returns the number of empty cells in a specified range of cells. |

| COUNTIF | Counts the cells based on specified condition. |

| COUNTIFS | Counts the cells based on specified conditions. |

| DATE | Returns the date based on given year, month, and day. |

| DATEVALUE | Converts a date string into date value. |

| DAY | Returns the day from the given date. |

| DAYS | Returns the number of days between two dates. |

| DECIMAL | Converts a text representation of a number in a given base into a decimal number. |

| DEGREES | Converts radians to degrees. |

| DOLLAR | Converts the number to currency formatted text. |

| EDATE | Returns a date with given number of months before or after the specified date. |

| EOMONTH | Returns the last day of the month that is a specified number of months before or after an initially supplied start date. |

| EVEN | Rounds a positive number up and negative number down to the nearest even integer. |

| EXACT | Checks whether a two text strings are exactly same and returns TRUE or FALSE. |

| EXP | Returns e raised to the power of the given number. |

| FACT | Returns the factorial of a number. |

| FIND | Returns the position of a string within another string, which is case sensitive. |

| FLOOR | Rounds a number down to the nearest multiple of a given factor. |

| HLOOKUP | Looks for a value in the top row of the array of values and then returns a value in the same column from a row in the array that you specify. |

| HOUR | Returns the number of hours in a specified time string. |

| IF | Returns value based on the given expression. |

| IFERROR | Returns value if no error found else it will return specified value. |

| IFS | Returns value based on the given multiple expressions. |

| INDEX | Returns a value of the cell in a given range based on row and column number. |

| INT | Rounds a number down to the nearest integer. |

| INTERCEPT | Calculates the point of the Y-intercept line via linear regression. |

| ISNUMBER | Returns true when the value parses as a numeric value. |

| LARGE | Returns the **k-th** largest value in a given array. |

| LEN | Returns a number of characters in a given string. |

| LN | Returns the natural logarithm of a number. |

| LOG | Returns the logarithm of a number to the base that you specify. |

| LOOKUP | Looks for a value in a one-row or one-column range, then returns a value from the same position in a second one-row or one-column range. |

| MATCH | Returns the relative position of a specified value in given range. |

| MAX | Returns the largest number of the given arguments. |

| MEDIAN | Returns the median of the given set of numbers. |

| MINUTE | Returns the number of minutes in a specified time string. |

| MIN | Returns the smallest number of the given arguments. |

| MOD | Returns a remainder after a number is divided by divisor. |

| MONTH | Returns the number of months in a specified date string. |

| NOT | Returns the inverse of a given logical expression. |

| NOW | Returns the current date and time. |

| ODD | Rounds a positive number up and negative number down to the nearest odd integer. |

| OR | Returns TRUE if any of the arguments are TRUE, otherwise returns FALSE. |

| PI | Returns the value of pi. |

| POWER | Returns the result of a number raised to power. |

| PRODUCT | Multiplies a series of numbers and/or cells. |

| RADIANS | Converts degrees into radians. |

| RAND | Returns a random number between 0 and 1. |

| RANDBETWEEN | Returns a random integer based on specified values. |

| ROUND | Rounds a number to the specified number of digits. |

| ROUNDDOWN | Rounds a number down, toward zero. |

| ROUNDUP | Rounds a number up, away from zero. |

| RSQ | Returns the square of the Pearson product moment correlation coefficient based on data points in *knowny's* and *knownx's*. |

| SECOND | Returns the number of seconds in a specified time string. |

| SMALL | Returns the **k-th** smallest value in a given array. |

| SLOPE | Returns the slope of the line from linear regression of the data points. |

- | SORT | Sorts the contents of a column, range, or array in ascending or descending order. |
- | SQRT | Returns the square root of a positive number. |
- | SUBTOTAL | Returns subtotal for a range using the given function number. |
- | SUM | Adds a series of numbers and/or cells. |
- | SUMIF | Adds the cells based on specified condition. |
- | SUMIFS | Adds the cells based on specified conditions. |
- | SUMPRODUCT | Returns the sum of the products of the corresponding array in given arrays. |
- | T | Checks whether a value is text or not and returns the text. |
- | TEXT | Converts the supplied value into text by using the user-specified format. |
- | TIME | Converts hours, minutes, seconds to the time formatted text. |
- | TODAY | Returns the current date. |
- | TRUNC | Truncates a supplied number to a specified number of decimal places. |
- | UNIQUE | Returns a unique values from a range or array. |
- | VLOOKUP | Looks for a specific value in the first column of a lookup range and returns a corresponding value from a different column within the same row. |

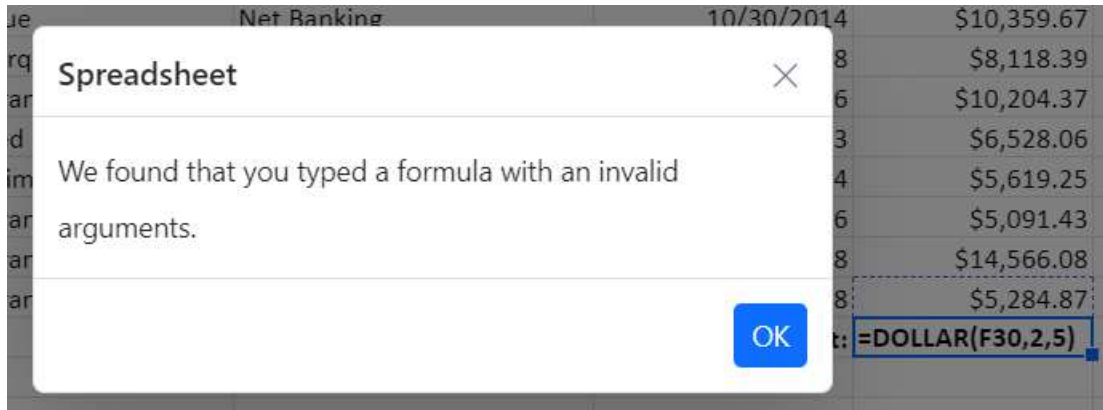
#### Formula Error Dialog

If you enter an invalid formula in a cell, an error dialog with an error message will appear. For instance, a formula with the incorrect number of arguments, a formula without parenthesis, etc.

| Error Message                                                                                                                                                  | Reason |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| -----                                                                                                                                                          | -----  |
| We found that you typed a formula with an invalid arguments   Occurs when passing an argument even though it wasn't needed.                                    |        |
| We found that you typed a formula with an empty expression   Occurs when passing an empty expression in the argument.                                          |        |
| We found that you typed a formula with one or more missing opening or closing parenthesis   Occurs when an open parenthesis or a close parenthesis is missing. |        |
| We found that you typed a formula which is improper   Occurs when passing a single reference but a range was needed.                                           |        |
| We found that you typed a formula with a wrong number of arguments   Occurs when the required arguments were not passed.                                       |        |
| We found that you typed a formula which requires 3 arguments   Occurs when the required 3 arguments were not passed.                                           |        |
| We found that you typed a formula with a mismatched quotes   Occurs when passing an argument with mismatched quotes.                                           |        |
| We found that you typed a formula with a circular reference   Occurs when passing a formula with circular cell reference.                                      |        |



| We found that you typed a formula which is invalid | Except in the cases mentioned above, all other errors will fall into this broad category. |



See Also

- [Editing](#)
- [Formatting](#)
- [Open](#)
- [Save](#)

### Formatting in Spreadsheet Component

Formatting options make your data easier to view and understand. The different types of formatting options in the Spreadsheet are,

- Number Formatting
- Text Formatting
- Cell Formatting

#### Number Formatting

Number formatting provides a type for your data in the Spreadsheet. Use the [allowNumberFormatting](#) property to enable or disable the number formatting option in the Spreadsheet. The different types of number formatting supported in Spreadsheet are,

| Types | Format |

|-----|-----|

| General(default) | NA |

| Number | 0.00 |

| Currency | \$#,##0.00 |

| Accounting | (\$ #,##0.00);(\$ (#,##0.00);(\$\* "-"??);(@\_) |

| ShortDate | mm-dd-yyyy |

| LongDate | dddd, mmmm dd, yyyy |

| Time | h:mm:ss AM/PM |

| Percentage | 0.00% |

| Fraction | # ?/? |

| Scientific | 0.00E+00 |

| Text | @ |

Number formatting can be applied in following ways,

- Using the `format` property in `cell`, you can set the desired format to each cell at initial load.
- Using the `numberFormat` method, you can set the number format to a cell or range of cells.
- Selecting the number format option from ribbon toolbar.

### Custom Number Formatting

Spreadsheet supports custom number formats to display your data as numbers, dates, times, percentages, and currency values. If the pre-defined number formats do not meet your needs, you can set your own custom formats using custom number formats dialog or `numberFormat` method.

The different types of custom number formatting supported in Spreadsheet are,

| Types | Format |

|-----|-----|

| General(default) | NA |

| Number | 0 |

| Number | 0.00 |

| Number | #,##0 |

| Number | #,##0.00 |

| Number | #,##0\_);(,##0) |

| Number | `#,##0\_);Red` |

| Number | #,##0.00\_);(,##0.00) |

| Number | `#,##0.00\_);Red` |

| Currency | \$#,##0\_);(\$#,##0) |

| Currency | `\$#,##0\_);Red` |

| Currency | \$#,##0.00\_);(\$#,##0.00) |

| Currency | `\$#,##0.00\_);Red` |

| Percentage | 0% |

| Percentage | 0.00% |

| Scientific | 0.00E+00 |

| Scientific | ##0.0E+0 |

| Fraction | # ?/? |  
 | Fraction | # ??/?? |  
 | ShortDate | dd-mm-yy |  
 | Custom | dd-mmm-yy |  
 | Custom | dd-mmm |  
 | Custom | mmm-yy |  
 | Custom | h:mm AM/PM |  
 | Custom | h:mm:ss AM/PM |  
 | Custom | h:mm |  
 | Custom | h:mm:ss |  
 | Custom | dd-mm-yy h:mm |  
 | Custom | mm:ss |  
 | Custom | mm:ss.0 |  
 | Text | @ |  
 | Custom | [h]:mm:ss |  
 | Accounting | (\$ #,##0);(\$ #,##0);(\$\* "-");(@\_) |  
 | Accounting | ( #,##0);( #,##0);(\* "-");(@\_) |  
 | Accounting | (\$ #,##0.00);(\$ #,##0.00);(\$\* "-"?);(@\_) |  
 | Accounting | ( #,##0.00);( #,##0.00);(\* "-"?);(@\_) |

Custom Number formatting can be applied in following ways,

- Using the `numberFormat` method, you can set your own custom number format to a cell or range of cells.
- Selecting the custom number format option from custom number formats dialog or type your own format in dialog input and then click apply button. It will apply the custom format for selected cells.

The following code example shows the number formatting in cell data.

### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).ShowFormulaBar(false).AllowInsert(false).AllowDelete(false)
.Sheets(sheet =>
{
 sheet.SelectedRange("U15").ShowGridLines(false).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A2").
 Add();
 }
})
```

```

 }).Rows(row =>
 {
 row.Height(35).CustomHeight(true).Cells(cell =>
 {
 cell.ColSpan(6).Value("Sales Team Summary").Style(new
 SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, VerticalAlign =
 VerticalAlign.Middle, TextAlign=TextAlign.Center, FontSize="16pt",
 Border="1px solid #e0e0e0", BackgroundColor="#EEEEEE", Color="#279377"
 }).Add();
 }).Add();
 row.Index(10).Cells(cell =>
 {
 cell.Index(1).Value("Total:").Style(new SpreadsheetCellStyle() {
 FontWeight = FontWeight.Bold, FontStyle= FontStyle.Italic }).Add();
 cell.Formula("=SUM(C3:C10)").Format("$#,##0.00").Add();
 cell.Formula("=SUM(D3:D10)").Format("_($* #,##0.00_);_($*
 (#,##0.00);_($* \"-\"??_);_(@_)").Add();
 cell.Formula("=SUM(E3:E10)").Format("_($* #,##0.00_);_($*
 (#,##0.00);_($* \"-\"??_);_(@_)").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(140).Add();
 column.Width(140).Add();
 column.Width(160).Add();
 column.Width(160).Add();
 column.Width(160).Add();
 column.Width(120).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
 backgroundColor: '#279377', textAlign: 'center', color: 'ffffff',
 borderBottom: '1px solid #e0e0e0' }, 'A2:F2');
 this.cellFormat({ borderTop: '1px solid #e0e0e0', backgroundColor:
 'EEEEEE' }, 'A11:F11');
 this.setBorder({ border: '1px solid #e0e0e0' }, 'A2:F11', 'Outer');
 // Applied Accounting format to the cells from C3 to E10 range.
 this.numberFormat('_($* #,##0.00_);_($* (#,##0.00);_($* "-
 ??_);_(@_) ', 'C3:E10');
 // Applied Percentage format to the cells from C3 to E11 range.
 this.numberFormat('0%', 'F3:F10');
 // applied the custom number format for cell form D3 to D10 range
 this.numberFormat(' [Red] [<=2000] $#,##0.00; [Blue] [>2000] $#,##0.00',
 'D3:D10');
 // applied the custom number format for cell from F3 to F10 range
 this.numberFormat('#,##0.00_); [Red] (#,##0.00) ', 'F3:F10');
 }
</script>

```

### NUMBERFORMATCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()

```

```

 {
 new { Salesperson= "Jeffrey Burke", RegionCovered=
"Oklahoma", February2019Sales= "28000", CostofSales= "2460",
January2019Sales= "21238", PercentChange = ".32"},
 new { Salesperson= "Amy Fernandez", RegionCovered= "North
Carolina", February2019Sales= "23138", CostofSales= "1521",
January2019Sales= "23212", PercentChange = "0"},
 new { Salesperson= "Mark Hayes", RegionCovered=
"Massachusetts", February2019Sales= "25092", CostofSales= "1521",
January2019Sales= "20454", PercentChange = ".23"},
 new { Salesperson= "Judith Ray", RegionCovered=
"California", February2019Sales= "21839", CostofSales= "1923",
January2019Sales= "24619", PercentChange = "-.11"},
 new { Salesperson= "Rany Graham", RegionCovered= "South
Carolina", February2019Sales= "23342", CostofSales= "2397",
January2019Sales= "20045", PercentChange = ".16"},
 new { Salesperson= "Christina Foster", RegionCovered=
"Delaware", February2019Sales= "23368", CostofSales= "1500",
January2019Sales= "17537", PercentChange = ".33"},
 new { Salesperson= "Judy Green", RegionCovered= "Texas",
February2019Sales= "21510", CostofSales= "1657", January2019Sales=
"17537", PercentChange = "-.14"},
 new { Salesperson= "Paula Hall", RegionCovered= "Virginia",
February2019Sales= "21314", CostofSales= "2418", January2019Sales=
"18082", PercentChange = ".18"},
 };
 ViewBag.DefaultData = data;
 return View();
 }
}

```

### Text and cell formatting

Text and cell formatting enhances the look and feel of your cell. It helps to highlight a particular cell or range of cells from a whole workbook. You can apply formats like font size, font family, font color, text alignment, border etc. to a cell or range of cells. Use the [allowCellFormatting](#) property to enable or disable the text and cell formatting option in Spreadsheet. You can set the formats in following ways,

- Using the `style` property, you can set formats to each cell at initial load.
- Using the `cellFormat` method, you can set formats to a cell or range of cells.
- You can also apply by clicking the desired format option from the ribbon toolbar.

### Fonts

Various font formats supported in the spreadsheet are font-family, font-size, bold, italic, strike-through, underline and font color.

### Text Alignment

You can align text in a cell either vertically or horizontally using the `textAlign` and `verticalAlign` property.

### Indents

To enhance the appearance of text in a cell, you can change the indentation of a cell content using `textIndent` property.

*Fill color*

To highlight cell or range of cells from whole workbook you can apply background color for a cell using `backgroundColor` property.

*Borders*

You can add borders around a cell or range of cells to define a section of worksheet or a table. The different types of border options available in the spreadsheet are,

| Types | Actions |

|-----|-----|

| Top Border | Specifies the top border of a cell or range of cells. |

| Left Border | Specifies the left border of a cell or range of cells. |

| Right Border | Specifies the right border of a cell or range of cells. |

| Bottom Border | Specifies the bottom border of a cell or range of cells. |

| No Border | Used to clear the border from a cell or range of cells. |

| All Border | Specifies all border of a cell or range of cells. |

| Horizontal Border | Specifies the top and bottom border of a cell or range of cells. |

| Vertical Border | Specifies the left and right border of a cell or range of cells. |

| Outside Border | Specifies the outside border of a range of cells. |

| Inside Border | Specifies the inside border of a range of cells. |

You can also change the color, size, and style of the border. The size and style supported in the spreadsheet are,

| Types | Actions |

|-----|-----|

| Thin | Specifies the `1px` border size (default). |

| Medium | Specifies the `2px` border size. |

| Thick | Specifies the `3px` border size. |

| Solid | Used to create the `solid` border (default). |

| Dashed | Used to create the `dashed` border. |

| Dotted | Used to create the `dotted` border. |

| Double | Used to create the `double` border. |

Borders can be applied in the following ways,

- Using the `border`, `borderLeft`, `borderRight`, `borderBottom` properties, you can set the desired border to each cell at initial load.
- Using the `setBorder` method, you can set various border options to a cell or range of cells.
- Selecting the border options from ribbon toolbar.

The following code example shows the style formatting in text and cells of the spreadsheet.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).ShowFormulaBar(false).AllowInsert(false).AllowDelete(false)
.AllowEditing(false).Sheets(sheet =>
{
 sheet.SelectedRange("U15").ShowGridLines(false).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A2").
Add();
 }).Rows(row =>
 {
 row.Height(40).CustomHeight(true).Cells(cell =>
 {
 cell.ColSpan(5).Value("Order Summary").Style(new
SpreadsheetCellStyle() { FontWeight = FontWeight.Bold, VerticalAlign =
VerticalAlign.Middle, TextAlign=TextAlign.Center, FontSize="18pt",
Border="1px solid #e0e0e0", BackgroundColor="#EEEEEE", Color="#279377"
}).Add();
 }).Add();
 row.Height(30).Cells(cell =>
 {
 cell.Index(2).Style(new SpreadsheetCellStyle() { TextAlign =
TextAlign.Right }).Add();
 }).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 row.Height(30).Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(200).Add();
 column.Width(110).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 }).Add();
}).Render()
<script>
function created() {
 // Setting common styles to table header cells
 this.cellFormat({ fontWeight: 'bold', fontSize: '12pt',
backgroundColor: '#279377', color: 'ffffff' }, 'A2:E2');
 // Setting common styles to whole table cells
 this.cellFormat({ verticalAlign: 'middle', fontFamily: 'Axettac
Demo' }, 'A2:E12');
 // Column wise styles setting
 this.cellFormat({ textAlign: 'center' }, 'A2:A12');
```

```

// Setting text-indent to 2 and 4 column
var style = { textAlign: 'left', textIndent: '8pt' };
this.cellFormat(style, 'B2:B12');
this.cellFormat(style, 'D2:D12');
this.cellFormat({ fontStyle: 'italic', textAlign: 'right' },
'C3:C12');
this.cellFormat({ textAlign: 'center' }, 'E2:E12');
// Applied border to range of cells using 'setBorder' method
this.setBorder({ borderLeft: '1px solid #e0e0e0', borderRight: '1px
solid #e0e0e0' }, 'A2:E2');
this.setBorder({ border: '1px solid #e0e0e0' }, 'A4:E11',
'Horizontal');
this.setBorder({ border: '1px solid #e0e0e0' }, 'A3:E12', 'Outer');
this.cellFormat({ color: '#10c469', text-decoration: 'line-through'
}, 'E3:E4');
this.cellFormat({ color: '#10c469', text-decoration: 'line-through'
}, 'E9');
this.cellFormat({ color: '#10c469', text-decoration: 'line-through'
}, 'E12');
this.cellFormat({ color: '#FFC107', text-decoration: 'underline' },
'E5');
this.cellFormat({ color: '#FFC107', text-decoration: 'underline' },
'E8');
this.cellFormat({ color: '#FFC107', text-decoration: 'underline' },
'E11');
this.cellFormat({ color: '#62c9e8' }, 'E6');
this.cellFormat({ color: '#62c9e8' }, 'E10');
this.cellFormat({ color: '#ff5b5b' }, 'E7');
}
</script>

```

### CELLFORMATCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { OrderId= "SF1001", Product= "Laptop Backpack (Blue)",
OrderedDate= "02/14/2014", OrderedBy= "Rahul Sharma", Shipment=
"Delivered"},
 new { OrderId= "SF1002", Product= "Oppo F1 S mobile back
cover", OrderedDate= "06/11/2014", OrderedBy= "Adi Pathak", Shipment=
"Delivered"},
 new { OrderId= "SF1003", Product= "Tupperware 4 bottle set",
OrderedDate= "07/27/2014", OrderedBy= "Himani Arora", Shipment=
"Pending"},
 new { OrderId= "SF1004", Product= "Tupperware Lunch box",
OrderedDate= "11/21/2014", OrderedBy= "Samuel Samson", Shipment=
"Shipped"},
 new { OrderId= "SF1005", Product= "Panasonic Hair Dryer",
OrderedDate= "06/23/2014", OrderedBy= "Neha", Shipment= "Cancelled"},
 new { OrderId= "SF1006", Product= "Philips LED 2 bulb set",
OrderedDate= "07/22/2014", OrderedBy= "Christine J", Shipment= "Pending"},
 new { OrderId= "SF1007", Product= "Moto G4 plus headphone",
OrderedDate= "02/04/2014", OrderedBy= "Shiv Nagar", Shipment=
"Delivered"},
 }
}

```



```

 new { OrderId= "SF1008", Product= "Lakme Eyeliner Pencil",
OrderedDate= "11/30/2014", OrderedBy= "Cherry", Shipment= "Shipped"},
 new { OrderId= "SF1009", Product= "Listerine mouthwash",
OrderedDate= "07/09/2014", OrderedBy= "Siddartha Mishra", Shipment=
"Pending"},
 new { OrderId= "SF1010", Product= "Protinex original",
OrderedDate= "10/31/2014", OrderedBy= "Ravi Chugh", Shipment=
"Delivered"},
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Limitations of Formatting

The following features are not supported in Formatting:

- Insert row/column between the formatting applied cells.
- Formatting support for row/column.

### Conditional Formatting

Conditional formatting helps you to format a cell or range of cells based on the conditions applied. You can enable or disable conditional formats by using the `allowConditionalFormat` property.

**Note:** \* The default value for the `allowConditionalFormat` property is `true`.

### Apply Conditional Formatting

You can apply conditional formatting by using one of the following ways,

- Select the conditional formatting icon in the Ribbon toolbar under the Home Tab.
- Using the `conditionalFormat` method to define the condition.
- Using the `conditionalFormats` in sheets model.

Conditional formatting has the following types in the spreadsheet,

### Highlight cells rules

Highlight cells rules option in the conditional formatting enables you to highlight cells with a preset color depending on the cell's value.

The following options can be given for the highlight cells rules as type,

**Note:** \* 'GreaterThan', 'LessThan', 'Between', 'EqualTo', 'ContainsText', 'DateOccur', 'Duplicate', 'Unique'.

The following preset colors can be used for formatting styles,

**Note:** \* "RedFT" - Light Red Fill with Dark Red Text,

<br/>\* "YellowFT" - Yellow Fill with Dark Yellow Text,

<br/>\* "GreenFT" - Green Fill with Dark Green Text,

<br/>\* "RedF" - Red Fill,

<br/>\* "RedT" - Red Text.

### *Top bottom rules*

Top bottom rules option in the conditional formatting allows you to apply formatting to the cells that satisfy a statistical condition with other cells in the range.

The following options can be given for the top bottom rules as type,

**Note:** \* 'Top10Items', 'Bottom10Items', 'Top10Percentage', 'Bottom10Percentage', 'BelowAverage', 'AboveAverage'.

### *Data Bars*

You can apply data bars to represent the data graphically inside a cell. The longest bar represents the highest value and the shorter bars represent the smaller values.

The following options can be given for the data bars as type,

**Note:** \* 'BlueDataBar', 'GreenDataBar', 'RedDataBar', 'OrangeDataBar', 'LightBlueDataBar', 'PurpleDataBar'.

### *Color Scales*

Using color scales, you can format your cells with two or three colors, where different color shades represent the different cell values. In the Green-Yellow-Red(GYR) Color Scale, the cell that holds the minimum value is colored as red. The cell that holds the median is colored as yellow, and the cell that holds the maximum value is colored as green. All other cells are colored proportionally.

The following options can be given for the color scales as type,

**Note:** \* 'GYRColorScale', 'RYGColorScale', 'GWRCColorScale', 'RWGColorScale', 'BWRColorScale', 'RWBCColorScale', 'WRColorScale', 'RWColorScale', 'GWColorScale', 'WGColorScale', 'GYColorScale', 'YGColorScale'.

### *Icon Sets*

Icon sets will help you to visually represent your data with icons. Every icon represents a range of values. In the Three Arrows(colored) icon, the green arrow icon represents the values greater than 67%, the yellow arrow icon represents the values between 33% to 67%, and the red arrow icon represents the values less than 33%.

The following options can be given for the icon sets as type,

**Note:** \* 'ThreeArrows', 'ThreeArrowsGray', 'FourArrowsGray', 'FourArrows', 'FiveArrowsGray', 'FiveArrows', 'ThreeTrafficLights1', 'ThreeTrafficLights2', 'ThreeSigns', 'FourTrafficLights', 'FourRedToBlack', 'ThreeSymbols', 'ThreeSymbols2', 'ThreeFlags', 'FourRating', 'FiveQuarters', 'FiveRating', 'ThreeTriangles', 'ThreeStars', 'FiveBoxes'.

### *Custom Format*

Using custom format for conditional formatting you can set cell styles like color, background color, font style, font weight and underline.

In the MAY and JUN columns, we have applied conditional formatting custom format.

**Note:** \* In the Conditional format, custom format supported for Highlight cells rules and Top bottom rules.

### *Clear Rules*

You can clear the defined rules by using one of the following ways,

- Using the “Clear Rules” option in the Conditional Formatting button of HOME Tab in the ribbon to clear the rule from selected cells.
- Using the `clearConditionalFormat` method to clear the defined rules.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).Created("create
d").Sheets(sheet =>
{
 sheet.Name("Car Sales Record").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).ConditionalFormats(
 new List<ConditionalFormat> {
 new ConditionalFormat() { Type =
ConditionalFormatType.GreaterThan, CFColor: CFColor.RedFT, Value= "7700" ,
Range = "B2:B9" },
 new ConditionalFormat() { Type =
ConditionalFormatType.Bottom10Items , CFColor = CFColor.YellowFT, Value =
"4", Range = "C2:C9" },
 new ConditionalFormat() { Type =
ConditionalFormatType.BlueDataBar , Range = "D2:D9" },
 }).Columns(column =>
 {
 column.Width(120).Index(1).Add();
 }).Add();
 }).Render()
<script>
function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:N1');
 this.conditionalFormat({ type: "RYGColorScale", range: 'E2:E9' });
 this.conditionalFormat({ type: "ThreeArrows", range: 'H2:H9' });
 //Custom format
 this.conditionalFormat({ type: 'Top10Items', value: '1',
 format: { style: { color: '#ffffff', backgroundColor: '#009999',
fontWeight: 'bold' } }, range: 'F2:F9' });
 this.conditionalFormat({ type: 'Bottom10Items', value: '1',
 format: { style: { color: '#ffffff', backgroundColor: '#c68d53',
fontWeight: 'bold' } }, range: 'G2:G9' });
}
</script>
```

### CONDITIONALFORMATTINGCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> conditionalFormatData = new List<object>()
 {
 new { EVModel= "BMW I3", JAN= "1224", FEB= "423", MAR=
"585", APR= "367", MAY= "729", JUN= "733", TOTAL= "=SUM(B2=G2)" },
 new { EVModel= "Tesla Model S", JAN= "975", FEB= "763", MAR=
"723", APR= "483", MAY= "983", JUN= "589", TOTAL= "=SUM(B3=G3)" },
 new { EVModel= "Chevrolet Volt", JAN= "113", FEB= "289",
MAR= "675", APR= "458", MAY= "391", JUN= "198", TOTAL= "=SUM(B4=G4)" },
 }
```

```

 new { EVModel= "Jaguar I-PACE", JAN= "78", FEB= "177", MAR=
"244", APR= "99", MAY= "312", JUN= "129", TOTAL= "=SUM(B5=G5)" },
 new { EVModel= "Tesla Model X", JAN= "978", FEB= "1108",
MAR= "1604", APR= "879", MAY= "1070", JUN= "1001", TOTAL= "=SUM(B6=G6)" },
 new { EVModel= "Nissan LEAF", JAN= "229", FEB= "978", MAR=
"1202", APR= "822", MAY= "135", JUN= "878", TOTAL= "=SUM(B7=G7)" },
 new { EVModel= "Honda Clarity EV", JAN= "671", FEB= "1302",
MAR= "466", APR= "989", MAY= "679", JUN= "891", TOTAL= "=SUM(B8=G8)" },
 new { EVModel= "Toyota Prius Prime", JAN= "978", FEB=
"1362", MAR= "1872", APR= "678", MAY= "900", JUN= "867", TOTAL=
"=SUM(B9=G9)" }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Limitations of Conditional formatting

The following features have some limitations in Conditional Formatting:

- Insert row/column between the conditional formatting.
- Conditional formatting with formula support.
- Copy and paste the conditional formatting applied cells.
- Custom rule support.

### See Also

- [Rows and columns](#)
- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)
- [Ribbon customization](#)

## Illustrations in Spreadsheet control

Illustrations helps you to insert a image, shapes and graphic objects in the Essential JS 2 spreadsheet.

### Image

Adding images to a spreadsheet can enhance the visual appeal and help convey information more clearly.

**Note:** \* The default value for [allowImage](#) property is `true`.

### Insert Image

You can insert the image by using one of the following ways,

- Selecting the Insert tab in the Ribbon toolbar, and then choose the Image tab.
- Use the `insertImage()` method programmatically.

The available parameters in `insertImage()` method are,

| Parameter | Type | Description |
|-----------|------|-------------|
|-----------|------|-------------|

|-----|-----|-----|

| images | **ImageModel** | Specifies the options to insert image in spreadsheet. |

| range(optional) | **string** | Specifies the range in spreadsheet. |

The available arguments in **ImageModel** are:

- **src**: Specifies the image source.
- **id**: Specifies image element id.
- **height**: Specifies the height of the image.
- **width**: Specifies the width of the image.
- **top**: Specifies the height of the image.
- **left**: Specifies the width of the image.

**Note:** \* In spreadsheet, you can add many types of image files, including IMAGE, JPG, PNG, GIF and JPEG files.

#### Delete Image

- If you want to delete the image, just select the image firstly, and then press the Delete key.
- Use the **deleteImage()** method programmatically.

The available parameters in **deleteImage()** method are,

| Parameter | Type | Description |

|-----|-----|-----|

| id | **string** | Specifies the id of the image element to be deleted. |

| range(optional) | **string** | Specifies the range in spreadsheet. |

#### Image Customization

Image feature allows you to view and insert a image in a spreadsheet and you can change the height and width of the image by resizing and move it to another position.

#### Height and Width

- You can change the height and width of the image by resizing.
- Use the **height** and **width** property in the **insertImage()** method programmatically.

#### Top and Left

- You can change the position of the image by drag and drop.
- Use the **top** and **left** property in the **insertImage()** method programmatically.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowFormulaBar(false).ShowRibbon(false)
.Created("created").Sheets(sheet =>
{
 sheet.Name("Employee Deatils").SelectedRange("B2").Rows(row =>
```

```

{
 row.Index(1).Height(30).Cells(cell =>
 {
 cell.Value(" Mark").Add();
 }).Add();
 row.Index(2).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("Id").Style(new SpreadsheetCellStyle() {
VerticalAlign = VerticalAlign.Bottom }).Add();
 cell.Index(3).Value(": 1001").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Bottom }).Add();
 }).Add();
 row.Index(3).Cells(cell =>
 {
 cell.Index(2).Value("Gender").Add();
 cell.Index(3).Value(": Male").Add();
 }).Add();
 row.Index(4).Cells(cell =>
 {
 cell.Index(2).Value("Contact Preference").Add();
 cell.Index(3).Value(": Email").Add();
 }).Add();
 row.Index(5).Cells(cell =>
 {
 cell.Index(2).Value("Email").Add();
 cell.Index(3).Value(": mark@gmail.com").Add();
 }).Add();
 row.Index(6).Cells(cell =>
 {
 cell.Index(2).Value("Date of Birth").Add();
 cell.Index(3).Value(": Jan 3, 1985").Add();
 }).Add();
 row.Index(7).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("Department").Add();
 cell.Index(3).Value(": IT").Add();
 }).Add();
 row.Index(8).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("IsActive").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Top }).Add();
 cell.Index(3).Value(": True").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Top }).Add();
 }).Add();
 row.Index(10).Height(30).Cells(cell =>
 {
 cell.Value(" Mary").Add();
 }).Add();
 row.Index(11).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("Id").Style(new SpreadsheetCellStyle() {
VerticalAlign = VerticalAlign.Bottom }).Add();
 cell.Index(3).Value(": 1002").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Bottom }).Add();
 }).Add();
 row.Index(12).Cells(cell =>
 {

```

```

 cell.Index(2).Value("Gender").Add();
 cell.Index(3).Value(": Female").Add();
 }).Add();
 row.Index(13).Cells(cell =>
 {
 cell.Index(2).Value("Contact Preference").Add();
 cell.Index(3).Value(": Phone").Add();
 }).Add();
 row.Index(14).Cells(cell =>
 {
 cell.Index(2).Value("Email").Add();
 cell.Index(3).Value(": mary@gmail.com").Add();
 }).Add();
 row.Index(15).Cells(cell =>
 {
 cell.Index(2).Value("Date of Birth").Add();
 cell.Index(3).Value(": Jan 3, 1985").Add();
 }).Add();
 row.Index(16).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("Department").Add();
 cell.Index(3).Value(": HR").Add();
 }).Add();
 row.Index(17).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("IsActive").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Top }).Add();
 cell.Index(3).Value(": True").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Top }).Add();
 }).Add();
 row.Index(19).Height(30).Cells(cell =>
 {
 cell.Value(" Nashi").Add();
 }).Add();
 row.Index(20).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("Id").Style(new SpreadsheetCellStyle() {
VerticalAlign = VerticalAlign.Bottom }).Add();
 cell.Index(3).Value(": 1003").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Bottom }).Add();
 }).Add();
 row.Index(21).Cells(cell =>
 {
 cell.Index(2).Value("Gender").Add();
 cell.Index(3).Value(": Female").Add();
 }).Add();
 row.Index(22).Cells(cell =>
 {
 cell.Index(2).Value("Contact Preference").Add();
 cell.Index(3).Value(": Email").Add();
 }).Add();
 row.Index(23).Cells(cell =>
 {
 cell.Index(2).Value("Email").Add();
 cell.Index(3).Value(": nashi@gmail.com").Add();
 }).Add();
 row.Index(24).Cells(cell =>

```

```

 {
 cell.Index(2).Value("Date of Birth").Add();
 cell.Index(3).Value(": Apr 11, 1986").Add();
 }).Add();
 row.Index(25).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("Department").Add();
 cell.Index(3).Value(": IT").Add();
 }).Add();
 row.Index(26).Height(40).Cells(cell =>
 {
 cell.Index(2).Value("IsActive").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Top }).Add();
 cell.Index(3).Value(": True").Style(new
SpreadsheetCellStyle() { VerticalAlign = VerticalAlign.Top }).Add();
 }).Add()).Add();
 }).Columns(column =>
 {
 column.Width(20).Add();
 column.Width(280).Add();
 column.Width(172).Add();
 column.Width(160).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.merge('B2:D2');
 this.merge('B11:D11');
 this.merge('B20:D20');
 this.cellFormat({ fontWeight: 'bold', verticalAlign: 'middle',
backgroundColor: '#1167b1', color: '#ffffff' }, 'B2');
 this.cellFormat({ fontWeight: 'bold', verticalAlign: 'middle',
backgroundColor: '#1167b1', color: '#ffffff' }, 'B11');
 this.cellFormat({ fontWeight: 'bold', verticalAlign: 'middle',
backgroundColor: '#1167b1', color: '#ffffff' }, 'B20');
 this.cellFormat({ fontWeight: 'bold' }, 'C3:C9');
 this.cellFormat({ fontWeight: 'bold' }, 'C12:C18');
 this.cellFormat({ fontWeight: 'bold' }, 'C21:C27');
 this.setBorder({ border: '1px solid #1167b1' }, 'B2:D9',
'Outer');
 this.setBorder({ border: '1px solid #1167b1' }, 'B11:D18',
'Outer');
 this.setBorder({ border: '1px solid #1167b1' }, 'B20:D27',
'Outer');
 }
</script>

```

### IMAGECONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```



### Limitations of Image

The following features have some limitations in Image:

- Corner resizing option in the image element.
- Copy and paste the external image.

### Chart

A chart is a graphical representation of data, that organizes and represents a set of numerical or qualitative data. It mostly displays the selected range of data in terms of x-axis and y-axis. You can use the [allowChart](#) property to enable or disable the chart functionality.

**Note:** \* The default value for the [allowChart](#) property is `true`.

### Types of chart

The following types of charts are available in the Spreadsheet.

**Note:** \* Column Chart

<br/>\* Bar Chart

<br/>\* Area Chart

<br/>\* Line Chart

<br/>\* Pie Chart

<br/>\* Scatter Chart

### Insert Chart

You can insert the chart by using one of the following ways,

- Select the chart icon in the Ribbon toolbar under the Insert Tab.
- Use the `insertChart()` method programmatically.

The available parameter in the `insertChart()` method is,

| Parameter | Type       | Description                                                 |
|-----------|------------|-------------------------------------------------------------|
| chart     | ChartModel | Specifies the options to insert a chart in the spreadsheet. |

The available arguments in the `ChartModel` are:

- `type`: Specifies the type of chart.
- `theme`: Specifies the theme of a chart.
- `isSeriesInRows`: Specifies to switch the row or a column.
- `range`: Specifies the selected range or specified range.
- `id`: Specifies the chart element id.
- `markerSettings`: Specifies the marker settings. The marker is used to provide information about the data points in the series and is currently only applicable to the line chart.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
```

```

{
 sheet.Name("Book Sales").Rows(row =>
 {
 row.Height(30).Cells(cell =>
 {
 cell.Value("Book Sales 2016-2020").Style(new
 SpreadsheetCellStyle() { BackgroundColor = "#357cd2", Color = "#fff",
 FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center, VerticalAlign =
 VerticalAlign.Middle }).Add();
 }).Add();
 }).Ranges(ranges =>
 {

ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A3").
Add();

 }).Columns(column =>
 {
 column.Width(110).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 }).Render()
<script>
 function created() {
 this.merge('A1:F1');
 this.cellFormat({ backgroundColor: '#357cd2', color: '#fff',
 fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
 this.numberFormat(getFormatFromType('Currency'), 'B4:F8');
 //Render Column chart
 this.insertChart([{type: 'Column', theme: 'Bootstrap5Dark', range:
 'A3:B6', id: 'column-chart' }]);
 //Render Line chart with Marker
 this.insertChart([{type: 'Line', range: 'A3:B6', markerSettings:
 {visible: true, shape: 'Circle', isFilled: false, size: 10, border: {width:
 2, color: '#3cb371'}}}, id: 'line-chart' }]);
 }
</script>

```

### CHARTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> chartData = new List<object>()
 {
 new { Book= "Classics", Year 2016= "19033", Year 2017=
 "78453", Year 2018= "24354", Year 2019= "18757", Year 2020= "34343" },
 new { Book= "Mystery", Year 2016= "50400", Year 2017=
 "82311", Year 2018= "131003", Year 2019= "19899", Year 2020= "42200" },
 new { Book= "Romance", Year 2016= "18002", Year 2017=
 "49529", Year 2018= "79567", Year 2019= "12302", Year 2020= "21277" },
 new { Book= "Sci-Fi & Fantasy", Year 2016= "10033", Year
 2017= "51200", Year 2018= "66211", Year 2019= "12899", Year 2020= "18779" },
 }
}

```

```

 new { Book= "Horror", Year 2016= "23454", Year 2017=
"78665", Year 2018= "81232", Year 2019= "19888", Year 2020= "20986" }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Delete Chart

- If you want to delete the chart, just select the chart, and then press the Delete key.
- Use the `deleteChart()` method programmatically.

The available parameter in the `deleteChart()` method is,

| Parameter | Type   | Description                                          |
|-----------|--------|------------------------------------------------------|
| id        | string | Specifies the id of the chart element to be deleted. |

### Chart Customization

Chart feature allows you to view and insert a chart in a spreadsheet, and you can change the height and width of the chart by resizing and moving it to another position.

- You can change the height and width of the chart by resizing.
- You can change the position of the chart by drag and drop.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
{
 sheet.Name("Book Sales").Rows(row =>
 {
 row.Height(30).Cells(cell =>
 {
 cell.Value("Book Sales 2016-2020").Style(new
SpreadsheetCellStyle() { BackgroundColor = "#357cd2", Color = "#fff",
FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center, VerticalAlign =
VerticalAlign.Middle }).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(7).Chart(new
List<Syncfusion.EJ2.Spreadsheet.Chart> { new
Syncfusion.EJ2.Spreadsheet.Chart() { Type =
Syncfusion.EJ2.Spreadsheet.ChartType.Column, Range = "A3:F8" } }).Add();
 }).Add();
 }).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A3").
Add();
 }).Columns(column =>
 {

```

```

 column.Width(110).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function created() {
 this.merge('A1:F1');
 this.cellFormat({ backgroundColor: '#357cd2', color: '#fff',
 fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
 this.numberFormat(getFormatFromType('Currency'), 'B4:F8');
 }
</script>

```

### CHARTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> chartData = new List<object>()
 {
 new { Book= "Classics", Year 2016= "19033", Year 2017=
"78453", Year 2018= "24354", Year 2019= "18757", Year 2020= "34343" },
 new { Book= "Mystery", Year 2016= "50400", Year 2017=
"82311", Year 2018= "131003", Year 2019= "19899", Year 2020= "42200" },
 new { Book= "Romance", Year 2016= "18002", Year 2017=
"49529", Year 2018= "79567", Year 2019= "12302", Year 2020= "21277" },
 new { Book= "Sci-Fi & Fantasy", Year 2016= "10033", Year
2017= "51200", Year 2018= "66211", Year 2019= "12899", Year 2020= "18779" },
 new { Book= "Horror", Year 2016= "23454", Year 2017=
"78665", Year 2018= "81232", Year 2019= "19888", Year 2020= "20986" }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Customization of line chart markers

Using the [actionBegin](#) event, you can change the shape, size, fill color, and border of the line chart marker. In the following example, you can see the modified marker appearance, such as shape and size, while creating the line chart with UI interaction.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Created("created").ActionBegin("onAct
ionBegin").Sheets(sheet =>
{
 sheet.Name("Book Sales").Rows(row =>
 {
 row.Height(30).Cells(cell =>
 {
 cell.Value("Book Sales 2016-2020").Style(new
SpreadsheetCellStyle() { BackgroundColor = "#357cd2", Color = "#fff",

```

```

FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center, VerticalAlign =
VerticalAlign.Middle }).Add();
 }).Add();
 }).Ranges(ranges =>
 {

ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A3").
Add();

 }).Columns(column =>
 {

 column.Width(110).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 }).Render()
<script>
 function created() {
 this.merge('A1:F1');
 this.cellFormat({ backgroundColor: '#357cd2', color: '#fff',
 fontWeight: 'bold', textAlign: 'center' }, 'A3:F3');
 this.numberFormat(getFormatFromType('Currency'), 'B4:F8');
 }
 function onActionBegin(args) {
 if (args.action === 'beforeInsertChart' &&
args.args.eventArgs.type.includes('Line')) {
 args.args.eventArgs.markerSettings.shape =
'Triangle';

 args.args.eventArgs.markerSettings.isFilled = false;
 args.args.eventArgs.markerSettings.size = 10;
 }
 }
}
</script>

```

### CHARTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> chartData = new List<object>()
 {
 new { Book= "Classics", Year 2016= "19033", Year 2017=
"78453", Year 2018= "24354", Year 2019= "18757", Year 2020= "34343" },
 new { Book= "Mystery", Year 2016= "50400", Year 2017=
"82311", Year 2018= "131003", Year 2019= "19899", Year 2020= "42200" },
 new { Book= "Romance", Year 2016= "18002", Year 2017=
"49529", Year 2018= "79567", Year 2019= "12302", Year 2020= "21277" },
 new { Book= "Sci-Fi & Fantasy", Year 2016= "10033", Year
2017= "51200", Year 2018= "66211", Year 2019= "12899", Year 2020= "18779" },
 new { Book= "Horror", Year 2016= "23454", Year 2017=
"78665", Year 2018= "81232", Year 2019= "19888", Year 2020= "20986" }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Limitations of Chart

The following features have some limitations in the Chart:

- Insert row/delete row between the chart data source will not reflect the chart.
- Copy/paste into the chart data source will not reflect the chart.
- Corner resizing option in chart element.

### See Also

- [Formatting](#)
- [Rows and columns](#)
- [Hyperlink](#)
- [Sorting](#)

## Data Binding in Spreadsheet Control

The Spreadsheet uses **DataManager**, which supports both RESTful JSON data services and local JavaScript object array binding to a range. The **dataSource** property can be assigned either with the instance of **DataManager** or JavaScript object array collection.

**Note:** To bind data to a cell, use **cell data binding** support.

### Local data

To bind local data to the Spreadsheet, you can assign a JavaScript object array to the **dataSource** property.

Refer to the following code example for local data binding.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
```

### LOCALDATACONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 }
```

```

 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

**Note:** The local data source can also be provided as an instance of the **DataManager**. By default, **DataManager** uses **JsonAdaptor** for local data-binding.

#### Remote data

To bind remote data to the Spreadsheet control, assign service data as an instance of **DataManager** to the **dataSource** property. To interact with remote data source, provide the service endpoint **url**.

Refer to the following code example for remote data binding.

#### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Sheets(sheet =>
{
 sheet.Name("Shipment Details").Ranges(ranges =>
 {

```

```

 ranges.ShowFieldAsHeader(false).StartCell("A2").Query("new
ej.data.Query().select(['OrderID', 'CustomerID', 'ShipName', 'ShipCity',
'ShipCountry', 'Freight']).take(200)").DataSource(dataManger =>
{
 dataManger.Url("https://services.syncfusion.com/js/production/api/Orders").C
rossDomain(true);
 }).Add();
 }).Rows(row =>
 {
 row.Cells(cell =>
 {
 cell.Value("Order ID").Add();
 cell.Value("Customer Name").Add();
 cell.Value("Freight").Add();
 cell.Value("Ship Name").Add();
 cell.Value("Ship City").Add();
 cell.Value("Ship Country").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(130).Add();
 column.Width(100).Add();
 column.Width(220).Add();
 column.Width(150).Add();
 column.Width(180).Add();
 }).Add();
 }).Render()

```

### REMOTEDATACONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

**Note:** By default, **DataManager** uses **ODataAdaptor** for remote data-binding.

#### *Binding with OData services*

**OData** is a standardized protocol for creating and consuming data. You can retrieve data from OData service using the DataManager. Refer to the following code example for remote Data binding using OData service.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
{
 sheet.Name("Order Details").Ranges(ranges =>
 {
 DataSource(dataManger =>
 {
 dataManger.Url("https://ej2services.syncfusion.com/production/web-
services/api/Orders").CrossDomain(true).Adaptor("ODataAdaptor");
 }).Add();
 }
}

```



```

 })..Columns(column =>
 {
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(280).Add();
 column.Width(180).Add();
 column.Width(80).Add();
 column.Width(180).Add();
 column.Width(180).Add();
 }).Add();
 }).Render()
<script>
 function dataBound() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 //Applies cell and number formatting to specified range of the
 active sheet
 spreadsheetObj.cellFormat({ fontWeight: 'bold', textAlign: 'center',
 verticalAlign: 'middle' },
 'A1:K1');
 }
</script>

```

### ODATACONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

### Web API

You can use WebApiAdaptor to bind spreadsheet with Web API created using OData endpoint.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
{
 sheet.Name("Order Details").Ranges(ranges =>
 {
 DataSource(dataManger =>
 {
 dataManger.Url("https://ej2services.syncfusion.com/production/web-
services/api/Orders").CrossDomain(true).Adaptor("WebApiAdaptor");
 }).Add();
 })..Columns(column =>
 {
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(80).Add();
 })
}

```

```

 column.Width(280).Add();
 column.Width(180).Add();
 column.Width(80).Add();
 column.Width(180).Add();
 column.Width(180).Add();
 }).Add();
}).Render()
<script>
 function dataBound() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 //Applies cell and number formatting to specified range of the
active sheet
 spreadsheetObj.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' },
 'A1:K1');
 }
</script>

```

### WEBAPICONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

### Cell data binding

The Spreadsheet control can bind the data to individual cell in a sheet . To achive this you can use the **value** property.

Refer to the following code example for cell data binding.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowRibbon(false).ShowFormulaBar(false).Sheets(sheet =>
{
 sheet.Name("Monthly Budget").SelectedRange("D13").Rows(row =>
 {
 row.Cells(cell =>
 {
 cell.Value("Category").Style(new SpreadsheetCellStyle() {
FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Planned cost").Style(new SpreadsheetCellStyle() {
FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 cell.Value("Actual cost").Style(new SpreadsheetCellStyle() {
FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center }).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Food").Add();
 cell.Value("$7000").Add();
 cell.Value("$8120").Add();
 }).Add();
 row.Cells(cell =>
 {

```

```

 cell.Value("Loan").Add();
 cell.Value("$1500").Add();
 cell.Value("$1500").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Medical").Add();
 cell.Value("$300").Add();
 cell.Value("$0").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Clothing").Add();
 cell.Value("$400").Add();
 cell.Value("$140").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Education").Add();
 cell.Value("$900").Add();
 cell.Value("$750").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Insurance").Add();
 cell.Value("$30").Add();
 cell.Value("$30").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Utilities").Add();
 cell.Value("$130").Add();
 cell.Value("$160").Add();
 }).Add();
 }).Columns(column =>
 {
 column.Width(110).Add();
 column.Width(115).Add();
 column.Width(110).Add();
 column.Width(100).Add();
 }).Add();
}).Render()

```

### CELLDATACONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

**Note:** The cell data binding also supports formula, style, number format, and more.

#### Dynamic data binding and Datasource change event

You can dynamically change the datasource of the spreadsheet by changing the `dataSource` property of the `range` object of the `sheet`. The `dataSourceChanged` event handler will be triggered when editing,

inserting, and deleting a row in the datasource range. This event will be triggered with a parameter named **action** which indicates the **edit**, **add** and **delete** actions for the respective ones.

The following table defines the arguments of the **dataSourceChanged** event.

| Property   | Type     | Description                                                                                                          |
|------------|----------|----------------------------------------------------------------------------------------------------------------------|
| action     | string   | Indicates the type of action such as <b>edit</b> , <b>add</b> , and <b>delete</b> performed in the datasource range. |
| data       | object[] | Modified data for <b>edit</b> action; New data for <b>add</b> action; Deleted data for <b>delete</b> action.         |
| rangeIndex | number   | Specifies the range index of the datasource.                                                                         |
| sheetIndex | number   | Specifies the sheet index of the datasource.                                                                         |

**Note:** For **add** action, the value for all the fields will be **null** in the data. In the case that you do not want the primary key field to be null which needs to be updated in the backend service, you can use **edit** action after updating the primary key field to update in the backend service.

For inserting a row at the end of the datasource range, you should insert a row below at the end of the range to trigger the **dataSourceChanged** event with action **add**.

### CSHTML

```
<div>
 <button id="changeDataBtn" class='e-btn'>Change Datasource</button>
</div>
@Html.EJS().Spreadsheet("spreadsheet").DataSourceChanged("dataSourceChanged")
.Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<div>
 <h4>Event Trace</h4>
 <div id="evt">
 <div style="height:173px;overflow: auto;min-
width: 250px;">
 <span id="EventLog" style="word-
break: normal;">
 </div>
 <button id="clearBtn" class='e-
btn'>Clear</button>
 </div>
</div>
<script>
document.getElementById('changeDataBtn').addEventListener('click', ()=> {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var itemData = [
 {
 'Item Name': 'Casual Shoes',
```

```

 'Date': '02/14/2019',
 'Time': '11:34:32 AM',
 'Quantity': 10,
 'Price': 20,
 'Amount': '=D2*E2',
 'Discount': 1,
 'Profit': 10
 },
 {
 'Item Name': 'Sports Shoes',
 'Date': '06/11/2019',
 'Time': '05:56:32 AM',
 'Quantity': 20,
 'Price': 30,
 'Amount': '=D3*E3',
 'Discount': 5,
 'Profit': 50
 }
];
spreadsheet.sheets[0].ranges[0].dataSource = itemData;
});
document.getElementById('clearBtn').addEventListener('click', () => {
 document.getElementById('EventLog').innerHTML = "";
});
function appendElement(html): void {
 var span = document.createElement("span");
 span.innerHTML = html;
 var log = document.getElementById('EventLog');
 log.insertBefore(span, log.firstChild);
}
function dataSourcechanged() {
 this.appendElement("Data source changed with" + " " +
args.action + " action<hr>"
);
}
</script>
<style>
#changeDataBtn {
 margin-bottom: 10px;
}
#EventLog b {
 color: #388e3c;
}
#evt {
 border: 1px solid #dcdcdc;
 padding: 10px;
}
hr {
 margin-top: 0;
 margin-bottom: 0;
}
</style>

```

**DYNAMICDATACONTROLLER.CS**

```
public IActionResult Index()
```

```

{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynnne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

- [Collaborative Editing](#)

### Filtering in Spreadsheet control

Filtering helps you to view specific rows in the spreadsheet by hiding the other rows. You can use the [allowFiltering](#) property to enable or disable filtering functionality.

**Note:** \* The default value for `allowFiltering` property is `true`.

By default, the `filter` module is injected internally into Spreadsheet to perform filtering.

#### Apply filter on UI

In the active Spreadsheet, select a range of cells to filter by value of the cell. The filtering can be done by any of the following ways:

- Select the filter item in the Ribbon toolbar.
- Right-click the sheet, select the filter item in the context menu.
- Use the `applyFilter()` method programmatically.
- Use `Ctrl + Shift + L` keyboard shortcut to apply the filter.

**Note:** \* Use `Alt + Up/Down` keyboard shortcut to open the filter dialog.

#### Filter by criteria

The `applyFilter()` method will apply the filter UI, based on the predicate and range given in the arguments.

**Note:** \* The `beforeFilter` event will be triggered before filtering the specified range.

**Note:** \* The `filterComplete` event will be triggered after the filter action is completed successfully.

The following code example shows `filter` functionality in the Spreadsheet control.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").DataBound("dataBound").AllowFiltering
(true).Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<script>
 function dataBound() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (spreadsheetObj.activeSheetIndex === 0) {
 var colors = ['Pink', 'Aquamarine', 'Blue'];
 var predicateList = []
 colors.forEach((color) => { predicateList.push({ field: 'C',
predicate: 'or', operator: 'equal', value: color }); })
 spreadsheetObj.applyFilter(predicateList);
 }
 }
</script>
```

**FILTERCONTROLLER.CS**

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gerner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```



### Filter by cell value

To apply a filter for a cell value, right-click the cell and choose filter -> **Filter By Selected Cell's Value** option from the menu. It applies the filter based on the value of the selected cell in the current sheet.

### Clear filter

After applying filter to a certain column, you may want to clear it to make all filtered rows visible again. It can be done in the following ways,

- Choose **Clear** option in ribbon toolbar under **Filter and Sort**. It clears the filters applied in the spreadsheet for all fields.
- Use the **clearFilter()** method programmatically, to clear the applied filters in spreadsheet for all fields.

### Clear filter on a field

After filtering, you can clear/reset the filter for a field alone. It can be done in the following ways,

- Click filter icon in the column's header and then choose **Clear Filter** option from the filter dialog.
- You can right-click on a filtered column cell and choose **Clear Filter from .** option from the context menu.
- Use the **clearFilter(field)** method programmatically, to clear the filter in a particular column.

### Reapply filter

When you want to reapply the filter after some changes happened in the rows. It can be done in the following ways,

- You can choose **Reapply** option in ribbon toolbar under **Filter and Sort** to reapply the filtered columns again.
- You can right-click on a filtered cell and choose **Reapply** option from the context menu. It reapplies the filters again in the Spreadsheet for all the fields.

### Known error validations

The following errors have been handled for filtering,

- **Out of range validation:** When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message **Select a cell or range inside the used range and try again** will be displayed. No filter will be performed if the range is invalid.

### Limitations

The following features have some limitations in Filter:

- Insert/delete row/column between the filter applied cells.
- Merge cells with filter.
- Copy/cut paste the filter applied cells.

## See Also

- [Sorting](#)
- [Hyperlink](#)
- [Undo Redo](#)

## Sorting in Spreadsheet control

Sorting helps arranging the data to a specific order in a selected range of cells. You can use the [allowSorting](#) property to enable or disable sorting functionality.

**Note:** \* The default value for `allowSorting` property is `true`.

By default, the `sort` module is injected internally into Spreadsheet to perform sorting.

## Sort by cell value

In the active Spreadsheet, select a range of cells to sort by cell value. The range sort can be done by any of the following ways:

- Select the sort item in the Ribbon toolbar and choose the ascending or descending item.
- Right-click the sheet, select the sort item in the context menu and choose the ascending/descending item.
- Use the `sort()` method programmatically.

The cell values can be sorted in the following orders:

- Ascending
- Descending

**Note:** \* Ascending is the default order for sorting.

The `sort()` method with empty arguments will sort the selected range by active cell's column as sort column in ascending order.

**Note:** \* The `beforeSort` event will be triggered before sorting the specified range.

`<br/>` \* The `sortComplete` event will be triggered after the sort action is completed successfully.

The following code example shows `sort` functionality in the Spreadsheet control.

**CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").AllowSorting(true).SortComplete("sort
Complete").DataBound("dataBound").BeforeSort("beforeSort").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
function dataBound() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
```

```

 if (spreadsheetObj.activeSheetIndex === 0) {
 spreadsheetObj.sort({ containsHeader: true }, 'A1:F15');
 }
 }
 function beforeSort(args) {
 //code here to handle sorting arguments.
 }
 function sortComplete(args) {
 spreadsheet.selectRange(args.range);
 // code here.
 }
}
</script>

```

### SORTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 }
}

```

```

 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Data contains header

You can specify whether the selected range of cells contains header. To specify, you need to set the `containsHeader` property to `true` and pass it as `sortOption` arguments of the `sort()` method.

**Note:** \* If the `containsHeader` property is not set and active cell column's first cell value type is differed from the second cell value type, the first row data in the range are marked as column headers.

You can also enable or disable this property using `beforeSort` event arguments,

```

`javascript
function beforeSort(args) {
args.sortOptions.containsHeader = true;
}
`

```

In the custom sort dialog, the `Data contains header` checkbox is checked on load. Thus, the default value for `containsHeader` is `true` in custom sort dialog.

### Case sensitive sort

The default sort functionality of Spreadsheet is a case insensitive sorting. When you want to perform sorting with case sensitive, you need to set the `caseSensitive` property to `true` and pass it as `sortOption` arguments of the `sort()` method.

Case sensitive sorting is applicable only for cells with alphabets. In ascending order sorting with case sensitive enabled, the cells with lower case text will be placed above the cells with upper case text.

**Note:** \* The default value for the `caseSensitive` property is `false`.

You can also enable or disable this property using `beforeSort` event arguments,

```

`javascript
function beforeSort (args) {
args.sortOptions.caseSensitive = true;
}
`

```

In the custom sort dialog, the `Case sensitive` checkbox is unchecked on load as the default value is `false`.

### Sort multiple columns

When you want to perform sorting on multiple columns, it can be done by any of the following ways:

- Select the **Custom sort...** menu item from the Ribbon toolbar item or context menu item.
- Use the `sort()` method programmatically by providing sort criteria.

**Note:** \* The current sorting functionality supports sorting based on cell values only.

#### Custom sort dialog

The custom sort dialog helps sorting multiple columns in the selected range by utilizing the rich UI. This dialog will be appeared while choosing the **Custom sort...** from the Ribbon item or context menu item. By default, sort criteria with the first column name from the selected range will be appeared in the dialog on initial load and it cannot be removed.

You can add multiple criteria using the **Add Column** button at the bottom of the dialog. Thus, multiple columns can be specified with different sort order. The newly added sort criteria items can be removed using the **delete** icons at the end of each items.

You can refer to the [Data contains header](#) topic to learn more about **Data contains header** checkbox. To learn more about **Case sensitive** checkbox, you can refer to [Case sensitive sort](#) topic.

#### Passing sort criteria manually

The multi-column sorting can also be performed manually by passing sort options to the `sort()` method programmatically. The `sortOption` have the following arguments:

- `sortDescriptors` – Sort criteria collection that holds the collection of field name, sort order, and `sortComparer`.
- `containsHeader` – Boolean argument that specifies whether the range has headers in it.
- `caseSensitive` – Boolean argument that specifies whether the range needs to consider case.

**Note:** \* All the arguments are optional.

\* When a `sortDescriptor` is specified without field, the field of the first `sortDescriptor` from the collection will be assigned from active cell's column name and others will be ignored. Hence, it will act as single column sorting.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").AllowSorting(true).SortComplete("sortComplete").DataBound("dataBound").BeforeSort("beforeSort").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
function dataBound() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var sortDescriptors = [{
 field: 'A',
```

```

 order: 'Ascending'
 },
 {
 field: 'B',
 order: 'Ascending'
 },
 {
 field: 'C',
 order: 'Descending'
 }
]];
 if (spreadsheetObj.activeSheetIndex === 0) {
 spreadsheetObj.sort({ sortDescriptors: sortDescriptors,
containsHeader: true }, 'A1:F15');
 }
}
function beforeSort(args) {
 //code here to handle sorting arguments.
}
function sortComplete(args) {
 spreadsheet.selectRange(args.range);
 // code here.
}
</script>

```

### PASSINGSORTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 }
}

```

```

 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Custom sort comparer

The `sortDescriptor` holds the `sortComparer` property, which is a function and it is used to customize the sort comparer for specific sort criteria. Each `sortDescriptor` can be customized using the custom sort comparer function.

By customizing sort comparer, you can define the sort action as desired.

**Note:** \* The `sortComparer` is an optional property of `sortDescriptor`.

For custom sort comparer example, refer to the [Sort a range by custom list](#) in the `how-to` section.

### Known error validations

The following errors have been handled for sorting,

- *Out of range validation:* When the selected range is not a used range of the active sheet, it is considered as invalid and the out of range alert with the message `Select a cell or range inside the used range and try again` will be displayed. No sort will be performed if the range is invalid.
- *Empty field validation:* When the sort criteria does not have a column selected (empty) in the custom sort dialog, it will become invalid, and an error message `Sort criteria column should not be empty` will be displayed on `OK` button click.
- *Duplicate field validation:* When the column names of added sort criteria are repeated more than once in the custom sort dialog, it will become invalid and an error message `is mentioned more than once. Duplicate columns must be removed` will be displayed on `OK` button click.

### Limitations

- Sorting is not supported with formula contained cells.

## See Also

- [Sort a range by custom list](#)
- [Hyperlink](#)
- [Filtering](#)
- [Undo Redo](#)

## Hyperlink in Spreadsheet control

Hyperlink is used to navigate to web links or cell reference within the sheet or to other sheets in Spreadsheet. You can use the [allowHyperlink](#) property to enable or disable hyperlink functionality.

**Note:** \* The default value for `allowHyperlink` property is `true`.

### Insert Link

You can insert a hyperlink in a worksheet cell for quick access to related information.

#### User Interface:

In the active spreadsheet, click the cell where you want to create a hyperlink. Insert hyperlink can be done by any of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the `Link` item.
- Right-click the cell and then click Hyperlink item in the context menu, or you can press Ctrl+K.
- Use the `addHyperlink()` method programmatically.

### Edit Hyperlink

You can change an existing hyperlink in your workbook by changing its destination or the text that is used to represent it.

#### User Interface:

In the active spreadsheet, Select the cell that contains the hyperlink that you want to change. Editing the hyperlink while opening the dialog can be done by any one of the following ways:

- Select the INSERT tab in the Ribbon toolbar and choose the `Link` item.
- Right-click the cell and then click Edit Hyperlink item in the context menu, or you can press Ctrl+K.

In the Edit Link dialog box, make the changes that you want, and click UPDATE.

### Remove Hyperlink

Performing this operation remove a single hyperlink without losing the display text.

#### User Interface:

In the active spreadsheet, click the cell where you want to remove a hyperlink. remove hyperlink can be done by any of the following ways:

- Right-click the cell and then click Remove Hyperlink item in the context menu.
- Use the `removeHyperlink()` method programmatically.



### How to change target attribute

There is an event named `beforeHyperlinkClick` which triggers only on clicking hyperlink. You can customize where to open the hyperlink by using the `target` property in the arguments of that event.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").BeforeHyperlinkClick("beforeHyperlinkClick").Sheets(sheet =>
{
 sheet.Name("PriceDetails").SelectedRange("D13").Rows(row =>
 {
 row.Cells(cell =>
 {
 cell.Value("Item Name").Add();
 cell.Value("Quantity").Add();
 cell.Value("Price").Add();
 cell.Value("Amount").Add();
 cell.Value("Stock Detail").Add();
 cell.Value("Website").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Casual Shoes").Add();
 cell.Value("10").Add();
 cell.Value("$20").Add();
 cell.Value("$200").Add();
 cell.Value("OUT OF STOCK").Add();
 cell.Value("Amazon").Hyperlink("www.amazon.com").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Sports Shoes").Add();
 cell.Value("20").Add();
 cell.Value("$30").Add();
 cell.Value("$600").Add();
 cell.Value("IN STOCK").Hyperlink("Stock!A3:B3").Add();
 cell.Value("Overstock").Hyperlink("www.overstock.com").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Formal Shoes").Add();
 cell.Value("20").Add();
 cell.Value("$15").Add();
 cell.Value("$300").Add();
 cell.Value("IN STOCK").Hyperlink("Stock!A2:B2").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("AliExpress").Hyperlink("www.aliexpress.com").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Sandals & Floaters").Add();
 cell.Value("15").Add();
 cell.Value("$20").Add();
 cell.Value("$300").Add();
 cell.Value("OUT OF STOCK").Add();
 cell.Value("Alibaba").Hyperlink("www.alibaba.com").Add();
 }).Add();
 });
});
```

```

 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Flip-Flops & Slippers").Add();
 cell.Value("30").Add();
 cell.Value("$10").Add();
 cell.Value("$300").Add();
 cell.Value("IN STOCK").Hyperlink("Stock!A4:B4").Add();
 cell.Value("Taobao").Hyperlink("www.taobao.com").Add();
 }).Add();
}).Columns(column =>
{
 column.Width(110).Add();
 column.Width(115).Add();
 column.Width(110).Add();
 column.Width(100).Add();
 column.Width(100).Add();
}).Add();
sheet.Name("Stock").SelectedRange("D13").Rows(row =>
{
 row.Cells(cell =>
 {
 cell.Value("Item Name").Add();
 cell.Value("Available Count").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Casual Shoes").Add();
 cell.Value("30").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Sports Shoes").Add();
 cell.Value("25").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Formal Shoes").Add();
 cell.Value("40").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Sandals & Floaters").Add();
 cell.Value("15").Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Value("Flip-Flops & Slippers").Add();
 cell.Value("30").Add();
 }).Add();
}).Columns(column =>
{
 column.Width(110).Add();
 column.Width(115).Add();
}).Add();
}).Render()

```

&lt;script&gt;

```
function beforeHyperlinkClick(args) {
 args.target = '_self'; //change target attribute
}
</script>
```

### **HYPERLINKCONTROLLER.CS**

```
public IActionResult Index()
{
 return View();
}
```

### Limitations

- Inserting hyperlink not supported for multiple ranges.

### See Also

- [Sorting](#)
- [Filtering](#)
- [Undo Redo](#)

### Clipboard in Spreadsheet control

The Spreadsheet provides support for the clipboard operations (cut, copy, and paste). Clipboard operations can be enabled or disabled by setting the [enableClipboard](#) property in Spreadsheet.

**Note:** By default, the `enableClipboard` property is true.

#### Cut

It is used to cut the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

#### User Interface:

Cut can be done in one of the following ways.

- Using Cut button in the Ribbon's HOME tab to perform cut operation.
- Using Cut option in the Context Menu.
- Using `Ctrl + X` | `Command + X` keyboard shortcut.
- Using the `cut` method.

#### Copy

It is used to copy the data from selected range of cells, rows or columns in a spreadsheet and make it available in the clipboard.

#### User Interface:

Copy can be done in one of the following ways.

- Using Copy button in the Ribbon's HOME tab to perform copy operation.
- Using Copy option in the Context Menu.

- Using **Ctrl + C** | **Command + C** keyboard shortcut.
- Using the **copy** method.

### Paste

It is used to paste the clipboard data to the selected range, rows or columns. You have the following options in Paste,

- **Paste Special** - You can paste the values with formatting.
- **Paste** - You can paste only the values without formatting.

It also performs for external clipboard operation. If you perform cut and paste, clipboard data will be cleared, whereas in copy and paste the clipboard contents will be maintained. If you perform paste inside the copied range, the clipboard data will be cleared.

### User Interface:

Paste can be done in one of the following ways.

- Using Paste button in the Ribbon's HOME tab to perform paste operation.
- Using Paste option in the Context Menu.
- Using **Ctrl + V** | **Command + V** keyboard shortcut.
- Using the **paste** method.

**Note:** If you use the Keyboard shortcut key for cut (**Ctrl + X**) | copy (**Ctrl + C**) from other sources, you should use **Ctrl + V** shortcut while pasting into the spreadsheet.

### CSHTML

```
@Html.EJS().DropDownButton("element").Content("Clipboard").Items((IEnumerable<object>)ViewBag.items).Select("itemSelect").Render()
@Html.EJS().Spreadsheet("spreadsheet").AllowScrolling(true).Created("createHandler").Sheets(sheet =>
{
 sheet.Name("Price Details").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(110).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 }).Add();
}).Render()
<script>
 document.getElementById("cutBtn").addEventListener('click', cut);
 document.getElementById("copyBtn").addEventListener('click', copy);
 document.getElementById("pasteBtn").addEventListener('click', paste);
 function createHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' }, 'A1:G1');
 }
}
```

```

function itemSelect(args) {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (args.item.text === 'Copy')
 spreadsheet.copy();
 if (args.item.text === 'Cut')
 spreadsheet.cut();
 if (args.item.text === 'Paste')
 spreadsheet.paste();
}
</script>

```

### CLIPBOARDCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 }
}

```

```

 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 ViewBag.DefaultData = data;
 return View();
}

```

### Prevent the paste functionality

The following example shows, how to prevent the paste action in spreadsheet. In [actionBegin](#) event, you can set **cancel** argument as false in paste request type.

### CSHTML

```

@Html.EJS().DropDownButton("element").Content("Clipboard").Items((IEnumerable<object>)ViewBag.items).Select("itemSelect").Render()
@Html.EJS().Spreadsheet("spreadsheet").AllowScrolling(true).Created("createHandler").ActionBegin("actionBeginHandler").Sheets(sheet =>
{
 sheet.Name("Price Details").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(110).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 }).Add();
}).Render()
<script>
 document.getElementById("cutBtn").addEventListener('click', cut);
 document.getElementById("copyBtn").addEventListener('click', copy);
 document.getElementById("pasteBtn").addEventListener('click', paste);
 function createHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' }, 'A1:G1');
 }
 // Triggers before the action begins.

```

```

function actionBeginHandler(pasteArgs) {
 // To cancel the paste action.
 if (pasteArgs.args.eventArgs.requestType === 'paste') {
 pasteArgs.args.eventArgs.cancel = true;
 }
}

function itemSelect(args) {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (args.item.text === 'Copy')
 spreadsheet.copy();
 if (args.item.text === 'Cut')
 spreadsheet.cut();
 if (args.item.text === 'Paste')
 spreadsheet.paste();
}
</script>

```

### CLIPBOARDCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gerner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Haring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 }
}

```

```

 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Copy"
 });
 items.Add(new
 {
 text = "Cut"
 });
 items.Add(new
 {
 text = "Paste"
 });
 ViewBag.items = items;
 ViewBag.DefaultData = data;
 return View();
}

```

## Limitations

- External clipboard is not fully supported while copying data from another source and pasting into a spreadsheet, it only works with basic supports (Values, Number, cell, and Text formatting).
- If you copy =SUM(A2,B2) and paste, the formula reference will change depending on the pasted cell address but we don't have support for nested formula(formula reference will be same).
- Clipboard is not supported with conditional formatting (values only pasting).
- We have limitation while copying the whole sheet data and pasting it into another sheet.

## Selection in Spreadsheet Control

Selection provides interactive support to highlight the cell, row, or column that you select. Selection can be done through Mouse, Touch, or Keyboard interaction. To enable selection, set `mode` as `Single` or `Multiple` in [selectionSettings](#). If you set `mode` to `None`, it disables the UI selection.

**Note:** The default value for `mode` in `selectionSettings` is `Multiple`.

You have the following options in Selection,

- Cell selection
- Row selection



- Column selection

### Cell selection

Cell selection is used to select a single or multiple cells. It can be performed using the `selectRange` method.

#### User Interface:

- Click on a cell to select it (or) use the `arrow` keys to navigate to it and select it.
- To select a range, select a cell, then use the left mouse button to select and drag over to other cells (or) use the `Shift + arrow` keys to select the range.
- To select non-adjacent cells and cell ranges, hold `Ctrl` and select the cells.

You can quickly locate and select specific cells or ranges by entering their names or cell references in the Name box, which is located to the left of the formula bar, and also select named or unnamed cells or ranges by using the Go To (`Ctrl+G`) command.

### Row selection

Row selection is used to select a single or multiple rows.

#### User Interface:

You can perform row selection in any of the following ways,

- By clicking the row header.
- To select multiple rows, select a row header with the left mouse button and drag over to other row headers (or) use the `Shift + arrow` keys to select multiple rows.
- To select non-adjacent rows, hold `Ctrl` and select the row header.
- You can also use the `selectRange` method for row selection.

The following sample shows the row selection in the spreadsheet, here selecting the 5th row using the `selectRange` method.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Created("createHandler").SelectionSettings(selectionSettings =>
{
 selectionSettings.Mode(SelectionMode.Multiple);
}).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 }).Add();
}).Render()
<script>
```

```
function createHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold' }, 'A1:D1');
 var colCount = this.getActiveSheet().colCount;
 this.selectRange(ej.spreadsheet.getRangeAddress([4, 0, 4,
colCount]));
}
</script>
```

### SELECTIONCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Expense Type= "Housing", Projected Cost= "7000",
Actual Cost= "7500", Difference= "-500" },
 new { Expense Type= "Transportation", Projected Cost= "500",
Actual Cost= "500", Difference= "0" },
 new { Expense Type= "Insurance", Projected Cost= "1000",
Actual Cost= "1000", Difference= "0" },
 new { Expense Type= "Food", Projected Cost= "2000", Actual
Cost= "1800", Difference= "200" },
 new { Expense Type= "Pets", Projected Cost= "300", Actual
Cost= "200", Difference= "100" },
 new { Expense Type= "Personel Care", Projected Cost= "500",
Actual Cost= "500", Difference= "0" },
 new { Expense Type= "Loan", Projected Cost= "1000", Actual
Cost= "1000", Difference= "0" },
 new { Expense Type= "Tax", Projected Cost= "200", Actual
Cost= "200", Difference= "0" },
 new { Expense Type= "Savings", Projected Cost= "1000",
Actual Cost= "900", Difference= "100" },
 new { Expense Type= "Total", Projected Cost= "13500", Actual
Cost= "13600", Difference= "-100" },
 };
 ViewBag.DefaultData = data;
 return View();
}
```

### Column selection

Column selection is used to select a single or multiple columns.

#### User Interface:

You can perform column selection in any of the following ways,

- By clicking the column header.
- To select multiple columns, select a column header with the left mouse button and drag over to other column headers (or) use the **Shift + arrow** keys to select the multiple columns.
- To select non-adjacent columns, hold **Ctrl** and select the column header.
- You can also use the **selectRange** method for row selection.

The following sample shows the column selection in the spreadsheet, here selecting the 3rd column using the `selectRange` method.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Created("createHandler").SelectionSettings(selectionSettings =>
{
 selectionSettings.Mode(SelectionMode.Multiple);
}).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 }).Add();
 }).Render()
<script>
 function createHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold' }, 'A1:D1');
 var rowCount = this.getActiveSheet().rowCount;
 this.selectRange(ej.spreadsheet.getRangeAddress([0, 2, rowCount,
2]));
 }
</script>
```

### SELECTIONCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Expense Type= "Housing", Projected Cost= "7000", Actual Cost= "7500", Difference= "-500" },
 new { Expense Type= "Transportation", Projected Cost= "500", Actual Cost= "500", Difference= "0" },
 new { Expense Type= "Insurance", Projected Cost= "1000", Actual Cost= "1000", Difference= "0" },
 new { Expense Type= "Food", Projected Cost= "2000", Actual Cost= "1800", Difference= "200" },
 new { Expense Type= "Pets", Projected Cost= "300", Actual Cost= "200", Difference= "100" },
 new { Expense Type= "Personel Care", Projected Cost= "500", Actual Cost= "500", Difference= "0" },
 new { Expense Type= "Loan", Projected Cost= "1000", Actual Cost= "1000", Difference= "0" },
 new { Expense Type= "Tax", Projected Cost= "200", Actual Cost= "200", Difference= "0" },
 new { Expense Type= "Savings", Projected Cost= "1000", Actual Cost= "900", Difference= "100" },
 }
}
```

```

 new { Expense Type= "Total", Projected Cost= "13500", Actual
Cost= "13600", Difference= "-100" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### How to remove selection in the spreadsheet

The following sample shows, how to remove the selection in the spreadsheet. Here changing the mode as **None** in [selectionSettings](#) to disable's the UI selection.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Created("createHandler").CellEdit("cellEdit").SelectionSettings(selectionSettings =>
{
 selectionSettings.Mode(SelectionMode.None);
}).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(130).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 }).Add();
 }).Render()
<script>
 function createHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold' }, 'A1:D1');
 }
 function cellEdit (args){
 args.cancel = true;
 }
</script>

```

### SELECTIONCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Expense Type= "Housing", Projected Cost= "7000",
Actual Cost= "7500", Difference= "-500" },
 new { Expense Type= "Transportation", Projected Cost= "500",
Actual Cost= "500", Difference= "0" },
 new { Expense Type= "Insurance", Projected Cost= "1000",
Actual Cost= "1000", Difference= "0" },
 new { Expense Type= "Food", Projected Cost= "2000", Actual
Cost= "1800", Difference= "200" },
 }
}

```

```

 new { Expense Type= "Pets", Projected Cost= "300", Actual
Cost= "200", Difference= "100" },
 new { Expense Type= "Personel Care", Projected Cost= "500",
Actual Cost= "500", Difference= "0" },
 new { Expense Type= "Loan", Projected Cost= "1000", Actual
Cost= "1000", Difference= "0" },
 new { Expense Type= "Tax", Projected Cost= "200", Actual
Cost= "200", Difference= "0" },
 new { Expense Type= "Savings", Projected Cost= "1000",
Actual Cost= "900", Difference= "100" },
 new { Expense Type= "Total", Projected Cost= "13500", Actual
Cost= "13600", Difference= "-100" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Limitations

- We have a limitation while performing the Select All(**ctrl + A**). You can do this only by clicking the Select All button at the top left corner.

## Scrolling in Spreadsheet control

Scrolling helps you to move quickly to different areas of the worksheet. It moves faster if we use horizontal and vertical scroll bars. Scrolling can be enabled by setting the [allowScrolling](#) as true.

**Note:** By default, the [allowScrolling](#) property is true.

You have the following options in Scrolling by using [scrollSettings](#).

- Finite scrolling.
- Virtual scrolling.

## Finite Scrolling

Finite scrolling supports two type of modes in scrolling. You can use the [isFinite](#) property in [scrollSettings](#) to specify the mode of scrolling.

- Finite - This mode does not create a new row/column when the scrollbar reaches the end. This can be achieved by setting the [isFinite](#) property as [true](#).
- Infinite - This mode creates a new row/column when the scrollbar reaches the end. This can be achieved by setting the [isFinite](#) property as [false](#).

**Note:** By Default, the [isFinite](#) property is [false](#).

## Virtual Scrolling

- Virtual scrolling allows you to load data that you require (load data based on viewport size) without buffering the entire huge database. You can set the [enableVirtualization](#) property in [scrollSettings](#) as [true](#).

In virtual scrolling `enableVirtualization` is set to true means, it allows you to load the spreadsheet data while scrolling.

**Note:** By Default, the `enableVirtualization` property is `true`.

#### User Interface:

You can scroll through the worksheet using one of the following ways,

- Using the `arrow` keys.
- Using the Horizontal and Vertical `scroll` bars.
- Using the `mouse` wheel.

#### Finite scrolling with defined rows and columns

If you want to perform scrolling with defined rows and columns, you must define `rowCount` and `colCount` in the `sheets` property and set `isFinite` as true and `enableVirtualization` as false in `scrollSettings`.

The following code example shows the finite scrolling with defined rows and columns in the spreadsheet. Here, we used `rowCount` as 20 and `colCount` as 20, after reaching the 20th row or 20th column you can't able to scroll.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").AllowScrolling(true).Created("createHandler").ScrollSettings(scrollSettings =>
{
 scrollSettings.IsFinite(true);
}).Sheets(sheet =>
{
 sheet.Name("Price Details").RowCount(9).ColCount(7).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(110).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 column.Width(110).Add();
 column.Width(92).Add();
 column.Width(96).Add();
 column.Width(96).Add();
 }).Add();
}).Render()
<script>
 function createHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' }, 'A1:F1');
 }
</script>
```

#### SCROLLINGCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynn Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Protection in Spreadsheet Control

Sheet protection helps you to prevent the users from modifying the data in the spreadsheet.

### Protect Sheet

Protect sheet feature helps you to prevent the unknown users from accidentally changing, editing, moving, or deleting data in a spreadsheet. And you can also protect the sheet with password. You can use the [isProtected](#) property to enable or disable the Protecting functionality.

**Note:** The default value for `isProtected` property is `false`.

By default in protected sheet, selecting, formatting, inserting, deleting functionalities are disabled. To enable some of the above said functionalities the `protectSettings` options are used in a protected spreadsheet.

The available `protectSettings` options in spreadsheet are,

| Options        | Uses                                  |
|----------------|---------------------------------------|
| -----          | -----                                 |
| Select Cells   | Used to perform Cell Selection.       |
| Format Cells   | Used to perform Cell formatting.      |
| Format Rows    | Used to perform Row formatting.       |
| Format Columns | Used to perform Column formatting.    |
| Insert Link    | Used to perform Hyperlink Insertions. |

**Note:** \* The default value for all `protectSettings` options are `false`.

By default, the `Protect Sheet` module is injected internally into the Spreadsheet to perform sheet protection function.

#### User Interface:

In the active Spreadsheet, the sheet protection can be done by any of the following ways:

- Select the Protect Sheet item in the Ribbon toolbar under the Data Tab, and then select your desired options.
- Right-click the sheet tab, select the Protect Sheet item in the context menu, and then select your desired options.
- Use the `protectSheet()` method programmatically.

The following example shows `Protect Sheet` functionality with password in the Spreadsheet control.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").DataBound("dataBound").Sheets(sheet =>
{
 sheet.Name("Budget").IsProtected(true).ProtectSettings(new
 SpreadsheetProtectSettings { SelectCells=false }).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.budgetData).StartCell("A1").Add();
 }).Columns(column =>
 {
```



```

 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 sheet.Name("Salary").Ranges(ranges =>
 {

ranges.DataSource((IEnumerable<object>) ViewBag.salaryData).StartCell("A1").Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 }).Render()
<script>
 function dataBound() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:D1');
 this.cellFormat({ fontWeight: 'bold', 'A11:D11' });
 spreadsheet.protectSheet(1, { selectCells: false, "syncfusion"); //
 protect sheet with password
 }
</script>

```

### PROTECTSHEETCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data1 = new List<object>()
 {
 new { ExpenseType= "Housing", ProjectedCost= "7000",
 ActualCost= "7500", Difference= "-500"},
 new { ExpenseType= "Transportation", ProjectedCost= "500",
 ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Insurance", ProjectedCost= "1000",
 ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Food", ProjectedCost= "2000",
 ActualCost= "1800", Difference= "200"},
 new { ExpenseType= "Pets", ProjectedCost= "300",
 ActualCost= "200", Difference= "100"},
 new { ExpenseType= "Personel Care", ProjectedCost= "500",
 ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Loan", ProjectedCost= "1000",
 ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Tax", ProjectedCost= "200", ActualCost=
 "200", Difference= "0"},
 new { ExpenseType= "Savings", ProjectedCost= "1000",
 ActualCost= "900", Difference= "100"},
 new { ExpenseType= "Total", ProjectedCost= "13500",
 ActualCost= "13600", Difference= "-100"},
 };
 List<object> data2 = new List<object>()

```

```

 {
 new { Earnings= "Basic", CreditAmount= "20000",
Deductions= "Provident Fund", DebitAmount= "2400"},
 new { Earnings= "HRA", CreditAmount= "8000", Deductions=
"ESI", DebitAmount= "0"},
 new { Earnings= "Special Allowance", CreditAmount= "25000",
Deductions= "Professional Tax", DebitAmount= "200"},
 new { Earnings= "Incentives", CreditAmount= "2000",
Deductions= "TDS", DebitAmount= "2750"},
 new { Earnings= "Bonus", CreditAmount= "1500", Deductions=
"Other Deduction", DebitAmount= "0"},
 new { Earnings= "Total Earnings", CreditAmount= "56500",
Deductions= "Total Deductions", DebitAmount= "5350"},
 };
 ViewBag.budgetData = data1;
 ViewBag.salaryData = data2;
 return View();
 }

```

### Limitations of Protect sheet

- Password encryption is not supported

### Unprotect Sheet

Unprotect sheet is used to enable all the functionalities that are already disabled in a protected spreadsheet.

#### User Interface:

In the active Spreadsheet, the sheet Unprotection can be done by any of the following ways:

- Select the **Unprotect Sheet** item in the Ribbon toolbar under the Data Tab.
- Right-click the sheet tab, select the **Unprotect Sheet** item in the context menu.
- Use the `unprotectSheet()` method programmatically.

### Unlock the particular cells in the protected sheet

In protected spreadsheet, to make some particular cell or range of cells are editable, you can use `lockCells()` method, with the parameter `range` and `isLocked` property as false.

### CSHTML

```

<button id="customBtn" class="e-btn"> Unlock cells</button>
@Html.EJS().Spreadsheet("spreadsheet").DataBound("dataBound").Sheets(sheet
=>
{
 sheet.Name("Budget").IsProtected(true).ProtectSettings(new
SpreadsheetProtectSettings { SelectCells = false }).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.budgetData).StartCell("A1").A
dd();
 }).Columns(column =>
 {
 column.Width(100).Add();
 });
}

```

```

 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 sheet.Name("Salary").Ranges(ranges =>
 {

ranges.DataSource((IEnumerable<object>)ViewBag.salaryData).StartCell("A1").Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 }).Render()
 @Html.EJS().Dialog("defaultDialog").Header("Spreadsheet").ShowCloseIcon(true)
 .Width("500px").Target("#spreadsheet").Buttons(ViewBag.DefaultButtons).Content(
 "'A1:F3' range of cells has been unlocked.")
 .IsModal(true).Visible(false).Render()
 <script>

 document.getElementById("customBtn").addEventListener('click',
 showAlert);
 function dataBound() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:D1');
 this.cellFormat({ fontWeight: 'bold' }, 'A11:D11');
 }
 function lockCells() {
 var spreadsheetObj =
 ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var dialogObj =
 ej.base.getComponent(document.getElementById('defaultDialog'), 'dialog');
 spreadsheetObj.lockCells('A1:F3', false);
 dialogObj.hide();
 }
 function showAlert() {
 var dialogObj =
 ej.base.getComponent(document.getElementById('defaultDialog'), 'dialog');
 dialogObj.show();
 }
 }
 </script>

```

### LOCKCELLCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data1 = new List<object>()
 {
 new { ExpenseType= "Housing", ProjectedCost= "7000",
 ActualCost= "7500", Difference= "-500"},
 new { ExpenseType= "Transportation", ProjectedCost= "500",
 ActualCost= "500", Difference= "0"},
 }
}

```

```

 new { ExpenseType= "Insurance", ProjectedCost= "1000",
ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Food", ProjectedCost= "2000",
ActualCost= "1800", Difference= "200"},
 new { ExpenseType= "Pets", ProjectedCost= "300",
ActualCost= "200", Difference= "100"},
 new { ExpenseType= "Personel Care", ProjectedCost= "500",
ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Loan", ProjectedCost= "1000",
ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Tax", ProjectedCost= "200", ActualCost=
"200", Difference= "0"},
 new { ExpenseType= "Savings", ProjectedCost= "1000",
ActualCost= "900", Difference= "100"},
 new { ExpenseType= "Total", ProjectedCost= "13500",
ActualCost= "13600", Difference= "-100"},
 };
 List<object> data2 = new List<object>()
 {
 new { Earnings= "Basic", CreditAmount= "20000",
Deductions= "Provident Fund", DebitAmount= "2400"},
 new { Earnings= "HRA", CreditAmount= "8000", Deductions=
"ESI", DebitAmount= "0"},
 new { Earnings= "Special Allowance", CreditAmount= "25000",
Deductions= "Professional Tax", DebitAmount= "200"},
 new { Earnings= "Incentives", CreditAmount= "2000",
Deductions= "TDS", DebitAmount= "2750"},
 new { Earnings= "Bonus", CreditAmount= "1500", Deductions=
"Other Deduction", DebitAmount= "0"},
 new { Earnings= "Total Earnings", CreditAmount= "56500",
Deductions= "Total Deductions", DebitAmount= "5350"},
 };
 List<DialogDialogButton> buttons = new
List<DialogDialogButton>() { };
 buttons.Add(new DialogDialogButton() { Click = "lockCells",
ButtonModel = new DefaultButtonModel() { content = "OK", isPrimary = true }
});

 ViewBag.DefaultButtons = buttons;
 ViewBag.budgetData = data1;
 ViewBag.salaryData = data2;
 return View();
}
public class DefaultButtonModel
{
 public string content { get; set; }
 public bool isPrimary { get; set; }
}

```

### Protect Workbook

Protect workbook feature helps you to protect the workbook so that users cannot insert, delete, rename, hide the sheets in the spreadsheet.

You can use the [password](#) property to protect workbook with password.

You can use the [isProtected](#) property to protect or unprotect the workbook without the password.

**Note:** The default value for `isProtected` property is `false`.

### User Interface:

In the active Spreadsheet, you can protect the worksheet by selecting the Data tab in the Ribbon toolbar and choosing the `Protect Workbook` item. Then, enter the password and confirm it and click on OK.

The following example shows `Protect Workbook` by using the `isProtected` property in the Spreadsheet control.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").IsProtected(true).DataBound("dataBound").Sheets(sheet =>
{
 sheet.Name("Budget").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.budgetData).StartCell("A1").Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function dataBound() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:D1');
 this.cellFormat({ fontWeight: 'bold' }, 'A11:D11');
 }
</script>
```

### PROTECTWORKBOOKCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ExpenseType= "Housing", ProjectedCost= "7000",
 ActualCost= "7500", Difference= "-500"},
 new { ExpenseType= "Transportation", ProjectedCost= "500",
 ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Insurance", ProjectedCost= "1000",
 ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Food", ProjectedCost= "2000",
 ActualCost= "1800", Difference= "200"},
 new { ExpenseType= "Pets", ProjectedCost= "300",
 ActualCost= "200", Difference= "100"},
 new { ExpenseType= "Personel Care", ProjectedCost= "500",
 ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Loan", ProjectedCost= "1000",
 ActualCost= "1000", Difference= "0"},
 }
```

```

 new { ExpenseType= "Tax", ProjectedCost= "200", ActualCost=
"200", Difference= "0"},
 new { ExpenseType= "Savings", ProjectedCost= "1000",
ActualCost= "900", Difference= "100"},
 new { ExpenseType= "Total", ProjectedCost= "13500",
ActualCost= "13600", Difference= "-100"},
 };
 ViewBag.budgetData = data;
 return View();
}

```

The following example shows **Protect Workbook** by using the [password](#) property in the Spreadsheet control. To unprotect the workbook, click the unprotect workbook button in the data tab and provide the password as syncfusion in the dialog box.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Password("syncfusion").DataBound("data
Bound").Sheets(sheet =>
{
 sheet.Name("Budget").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.budgetData).StartCell("A1").A
dd();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function dataBound() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:D1');
 this.cellFormat({ fontWeight: 'bold'}, 'A11:D11');
 }
</script>

```

### PASSWORDCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ExpenseType= "Housing", ProjectedCost= "7000",
ActualCost= "7500", Difference= "-500"},
 new { ExpenseType= "Transportation", ProjectedCost= "500",
ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Insurance", ProjectedCost= "1000",
ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Food", ProjectedCost= "2000",
ActualCost= "1800", Difference= "200"},
 }
}

```

```

 new { ExpenseType= "Pets", ProjectedCost= "300",
ActualCost= "200", Difference= "100"},
 new { ExpenseType= "Personel Care", ProjectedCost= "500",
ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Loan", ProjectedCost= "1000",
ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Tax", ProjectedCost= "200", ActualCost=
"200", Difference= "0"},
 new { ExpenseType= "Savings", ProjectedCost= "1000",
ActualCost= "900", Difference= "100"},
 new { ExpenseType= "Total", ProjectedCost= "13500",
ActualCost= "13600", Difference= "-100"},
 };
 ViewBag.budgetData = data;
 return View();
}

```

### Unprotect Workbook

Unprotect Workbook is used to enable the insert, delete, rename, move, copy, hide or unhide sheets feature in the spreadsheet.

#### User Interface:

In the active Spreadsheet, the workbook Unprotection can be done in any of the following ways:

- Select the **Unprotect Workbook** item in the Ribbon toolbar under the Data Tab and provide the valid password in the dialog box.

#### See Also

- [Hyperlink](#)

### Rows and columns in Spreadsheet control

Spreadsheet is a tabular format consisting of rows and columns. The intersection point of rows and columns are called as cells. The list of operations that you can perform in rows and columns are,

- Insert
- Delete
- Show and Hide

#### Insert

You can insert rows or columns anywhere in a spreadsheet. Use the [allowInsert](#) property to enable or disable the insert option in Spreadsheet.

#### Row

The rows can be inserted in the following ways,

- Using **insertRow** method, you can insert the rows once the component is loaded.
- Using context menu, insert the empty rows in the desired position.

The following code example shows the options for inserting rows in the spreadsheet.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).ShowFormulaBar(false).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("B1").
 Add();

 }).Columns(column =>
 {
 column.Width(20).Add();
 column.Width(90).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 }).Render()
<script>
 function created() {
 // Rows model that is going to insert dynamically
 var rowsModel = [{
 index: 9, // Need to specify the index for the first row
 collection, the specified rows will be inserted in this index.
 cells: [{ value: '' }, { value: '8' }, { value: 'Northwoods
Cranberry Sauce' }, { value: '3' }, { value: '12 - 12 oz jars' },
 { value: '40.00' }, { value: '6' }, { value: 'false' }]
 },
 {
 cells: [{ value: '' }, { value: '9' }, { value: 'Mishi Kobe
Niku' }, { value: '4' }, { value: '18 - 500 g pkgs.' }, { value: '97.00' },
 { value: '29' }, { value: 'true' }]
 }
];
 // Applies style formatting before inserting the rows
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'B1:H1');
 // inserting a empty row at 0th index
 this.insertRow();
 // inserting 2 rows at the 9th index with data
 this.insertRow(rowsModel);
 // Applies style formatting after the rows are inserted
 this.cellFormat({ textAlign: 'center' }, 'B3:B12');
 this.cellFormat({ textAlign: 'center' }, 'D3:D12');
 this.cellFormat({ textAlign: 'center' }, 'F3:H12');
 }
</script>
```

### INSERTROWCONTROLLER.CS

```
public IActionResult Index()
```



```

{
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
"1", QuantityPerUnit= "10 boxes x 20 bags", UnitPrice= "18.00",
UnitsInStock= "39", Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
"1", QuantityPerUnit= "24 - 12 oz bottles", UnitPrice= "19.00",
UnitsInStock= "17", Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitPrice=
"10.00", UnitsInStock= "13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
UnitPrice= "22.00", UnitsInStock= "53", Discontinued= "true" },
 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", 'Unit Price',
UnitPrice= "21.35", UnitsInStock= "0", Discontinued= "true" },
 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars", UnitPrice=
"25.00", UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitPrice= "30.00", UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "10", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitPrice= "21.00",
UnitsInStock= "22", Discontinued= "false" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Column

The columns can be inserted in the following ways,

- Using `insertColumn` method, you can insert the columns once the component is loaded.
- Using context menu, insert the empty columns in the desired position.

The following code example shows the options for inserting columns in the spreadsheet.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).ShowFormulaBar(false).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A2").
Add();

 }).Columns(column =>
 {
 column.Width(90).Add();
 column.Width(220).Add();
 })
}
)

```

```

 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function created() {
 // Cells model that you are going to update in the inserted 5th
 column dynamically
 var cellsModel = [{ value: 'UnitPrice', style: { fontWeight: 'bold',
textAlign: 'center' } }, { value: '18.00' },
 { value: '19.00' }, { value: '10.00' }, { value: '22.00' }, { value:
'21.35' }, { value: '25.00' }, { value: '30.00' },
 { value: '21.00' }, { value: '40.00' }, { value: '97.00' }];
 // Applies style formatting before inserting the column
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A2:G2');
 // inserting a empty column at 0th index
 this.insertColumn();
 // inserting 1 column at the 5th index with column model
 this.insertColumn([{ index: 5, width: 90 }]);
 var rowIndex = 1;
 // Updating the 5th column data
 cellsModel.forEach((cell) => {
 this.updateCell(cell, ej.spreadsheet.getCellAddress(rowIndex,
5)); rowIndex++;
 });
 // Applies style formatting after the columns are inserted
 this.cellFormat({ textAlign: 'center' }, 'B3:B12');
 this.cellFormat({ textAlign: 'center' }, 'D3:D12');
 this.cellFormat({ textAlign: 'center' }, 'F3:H12');
 }
</script>

```

### INSERTCOLUMNCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
"1", QuantityPerUnit= "10 boxes x 20 bags", UnitsInStock= "39",
Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
"1", QuantityPerUnit= "24 - 12 oz bottles", UnitsInStock= "17",
Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitsInStock=
"13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
UnitsInStock= "53", Discontinued= "true" },
 }
}

```

```

 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", 'Unit Price",
UnitsInStock= "0", Discontinued= "true" },
 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars",
UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "8", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitsInStock= "22",
Discontinued= "false" },
 new { ProductId= "9", ProductName= "Northwoods Cranberry
Sauce", SupplierId= "3", QuantityPerUnit= "12 - 12 oz jars",
UnitsInStock= "6", Discontinued= "false" },
 new { ProductId= "10", ProductName= "Mishi Kobe Niku",
SupplierId= "4", QuantityPerUnit= "18 - 500 g pkgs.", UnitsInStock= "29",
Discontinued= "false" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Delete

Delete support provides an option for deleting the rows and columns in the spreadsheet. Use [allowDelete](#) property to enable or disable the delete option in Spreadsheet.

The rows and columns can be deleted dynamically in the following ways,

- Using `delete` method, you can delete the loaded rows and columns.
- Using context menu, you can delete the selected rows and columns.

The following code example shows the delete operation of rows and columns in the spreadsheet.

## CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").AllowDelete(true).Created("created").
ShowRibbon(false).ShowFormulaBar(false).Sheets(sheet =>
{
 sheet.Name("Sheet1").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(90).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 sheet.Name("Sheet2").Ranges(ranges =>

```

```

{
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();

}).Columns(column =>
{
 column.Width(90).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
}).Add();
}).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:H1');
 // deleting the rows from 8th to 10th index. To delete row, the
 third argument of enum type is passed as 'Row', the last argument specifies
 the sheet name or index in which the delete operation will perform. By
 default, active sheet will be considered. It is applicable only for model
 type Row and Column.
 this.delete(8, 10, 'Row', 0); // startIndex, endIndex, Row, sheet
 index
 // deleting the 2nd and 5th indexed columns
 this.delete(2, 2, 'Column', 'Sheet2');
 this.delete(5, 5, 'Column');
 this.delete(0, 0, "Sheet"); // delete the first sheet. sheet index
 starts from 0
 // Applies style formatting after deleted the rows and columns
 this.cellFormat({ textAlign: 'center' }, 'A2:A8');
 this.cellFormat({ textAlign: 'center' }, 'D2:G8');
 }
</script>

```

### DELETEROWCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
 "1", QuantityPerUnit= "10 boxes x 20 bags", UnitPrice= "18.00",
 UnitsInStock= "39", Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
 "1", QuantityPerUnit= "24 - 12 oz bottles", UnitPrice= "19.00",
 UnitsInStock= "17", Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
 SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitPrice=
 "10.00", UnitsInStock= "13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
 Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
 UnitPrice= "22.00", UnitsInStock= "53", Discontinued= "true" },
 }
}

```

```

 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", Unit Price",
UnitPrice= "21.35", UnitsInStock= "0", Discontinued= "true" },
 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars", UnitPrice=
"25.00", UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitPrice= "30.00", UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "8", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitPrice= "21.00",
UnitsInStock= "22", Discontinued= "false" },
 new { ProductId= "9", ProductName= "Northwoods Cranberry
Sauce", SupplierId= "3", QuantityPerUnit= "12 - 12 oz jars", UnitPrice=
"21.00", UnitsInStock= "6", Discontinued= "false" },
 new { ProductId= "10", ProductName= "Mishi Kobe Niku",
SupplierId= "4", QuantityPerUnit= "18 - 500 g pkgs.", UnitPrice= "21.00",
UnitsInStock= "29", Discontinued= "true" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Hide and show

You can show or hide the rows and columns in the spreadsheet through property binding, method, and context menu.

#### Row

The rows can be hidden or shown through the following ways,

- Using `hidden` property in row, you can hide/show the rows at initial load.
- Using `hideRow` method, you can hide the rows by specifying the start and end row index, set the last argument `hide` as `false` to unhide the hidden rows.
- Right-click on the row header and select the desired option from context menu

#### Column

The columns can be hidden or shown through following ways,

- Using `hidden` property in columns, you can hide/show the columns at initial load.
- Using `hideColumn` method, you can hide the columns by specifying the start and end column index, set the last argument `hide` as `false` to unhide the hidden columns.
- Right-click on the column header and select the desired option from context menu

The following code example shows the hide/show rows and columns operation in the spreadsheet.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ShowSheetTabs(false).Created("created")
.ShowRibbon(false).ShowFormulaBar(false).Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {

```

```

 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();

 }).Rows(row =>
 {
 row.Index(2).Hidden(true).Add();
 row.Hidden(true).Add();
 }).Columns(column =>
 {
 column.Width(90).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 }).Render()
<script>
 function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
 'A1:H1');
 // Unhide the 2nd index hidden column
 this.hideColumn(1, 1, false);
 // Unhide the 3rd index hidden row
 this.hideRow(3, 3, false);
 // Hiding the 6th index column
 this.hideColumn(6);
 // Hiding the 8th and 9th index row
 this.hideRow(8, 9);
 this.cellFormat({ textAlign: 'center' }, 'D2:H11');
 }
</script>

```

### SHOWHIDECONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
 "1", QuantityPerUnit= "10 boxes x 20 bags", UnitPrice= "18.00",
 UnitsInStock= "39", Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
 "1", QuantityPerUnit= "24 - 12 oz bottles", UnitPrice= "19.00",
 UnitsInStock= "17", Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
 SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitPrice=
 "10.00", UnitsInStock= "13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
 Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
 UnitPrice= "22.00", UnitsInStock= "53", Discontinued= "true" },
 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
 Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", 'Unit Price',
 UnitPrice= "21.35", UnitsInStock= "0", Discontinued= "true" },
 }
}

```

```

 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars", UnitPrice=
"25.00", UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitPrice= "30.00", UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "8", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitPrice= "21.00",
UnitsInStock= "22", Discontinued= "false" },
 new { ProductId= "9", ProductName= "Northwoods Cranberry
Sauce", SupplierId= "3", QuantityPerUnit= "12 - 12 oz jars", UnitPrice=
"21.00", UnitsInStock= "6", Discontinued= "false" },
 new { ProductId= "10", ProductName= "Mishi Kobe Niku",
SupplierId= "4", QuantityPerUnit= "18 - 500 g pkgs.", UnitPrice= "21.00",
UnitsInStock= "29", Discontinued= "true" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Size

You can change the size of rows and columns in the spreadsheet by using `setRowsHeight` and `setColumnsWidth` methods.

### Row

You can change the height of single or multiple rows by using the `setRowsHeight` method.

You can provide the following type of ranges to the method:

- Single row range: ['2:2']
- Multiple rows range: ['1:100']
- Multiple rows with discontinuous range: ['1:10', '15:25', '30:40']
- Multiple rows with different sheets: [Sheet1!1:50, 'Sheet2!1:50', 'Sheet3!1:50']

The following code example shows how to change the height for single/multiple rows in the spreadsheet.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<script>
function created() {
 // To change height for single row
 this.setRowsHeight(40, ['2']);
 // To change height for multiple rows
 this.setRowsHeight(40, ['4:8', '10:12']);
}

```

```
</script>
```

**ROWHEIGHTCONTROLLER.CS**

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
"1", QuantityPerUnit= "10 boxes x 20 bags", UnitPrice= "18.00",
UnitsInStock= "39", Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
"1", QuantityPerUnit= "24 - 12 oz bottles", UnitPrice= "19.00",
UnitsInStock= "17", Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitPrice=
"10.00", UnitsInStock= "13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
UnitPrice= "22.00", UnitsInStock= "53", Discontinued= "true" },
 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", Unit Price=
UnitPrice= "21.35", UnitsInStock= "0", Discontinued= "true" },
 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars", UnitPrice=
"25.00", UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitPrice= "30.00", UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "8", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitPrice= "21.00",
UnitsInStock= "22", Discontinued= "false" },
 new { ProductId= "9", ProductName= "Northwoods Cranberry
Sauce", SupplierId= "3", QuantityPerUnit= "12 - 12 oz jars", UnitPrice=
"21.00", UnitsInStock= "6", Discontinued= "false" },
 new { ProductId= "10", ProductName= "Mishi Kobe Niku",
SupplierId= "4", QuantityPerUnit= "18 - 500 g pkgs.", UnitPrice= "21.00",
UnitsInStock= "29", Discontinued= "true" },
 };
 ViewBag.DefaultData = data;
 return View();
}
```

*Column*

You can change the width of single or multiple columns by using the `setColumnsWidth` method.

You can provide the following type of ranges to the method:

- Single column range: ['F:F']
- Multiple columns range: ['A:F']
- Multiple columns with discontinuous range: ['A:C', 'G:I', 'K:M']
- Multiple columns with different sheets: [Sheet1!A:H, 'Sheet2!A:H', 'Sheet3!A:H']



The following code example shows how to change the width for single/multiple columns in the spreadsheet.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Created("created").Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();

 }).Add();
}).Render()
<script>
function created() {
 // To change width of single column
 this.setColumnsWidth(100, ['F']);
 // To change width of multiple columns
 this.setColumnsWidth(120, ['A:C', 'G:I', 'K:M']);
}
</script>
```

### COLUMNWIDTHCONTROLLER.CS

```
public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
"1", QuantityPerUnit= "10 boxes x 20 bags", UnitPrice= "18.00",
UnitsInStock= "39", Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
"1", QuantityPerUnit= "24 - 12 oz bottles", UnitPrice= "19.00",
UnitsInStock= "17", Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitPrice=
"10.00", UnitsInStock= "13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
UnitPrice= "22.00", UnitsInStock= "53", Discontinued= "true" },
 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", Unit Price=
UnitPrice= "21.35", UnitsInStock= "0", Discontinued= "true" },
 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars", UnitPrice=
"25.00", UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitPrice= "30.00", UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "8", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitPrice= "21.00",
UnitsInStock= "22", Discontinued= "false" },
 new { ProductId= "9", ProductName= "Northwoods Cranberry
Sauce", SupplierId= "3", QuantityPerUnit= "12 - 12 oz jars", UnitPrice=
"21.00", UnitsInStock= "6", Discontinued= "false" },
```

```

 new { ProductId= "10", ProductName= "Mishi Kobe Niku",
SupplierId= "4", QuantityPerUnit= "18 - 500 g pkgs.", UnitPrice= "21.00",
UnitsInStock= "29", Discontinued= "true" },
 };
 ViewBag.DefaultData = data;
 return View();
 }
}

```

### Changing text in column headers

Using the [beforeCellRender](#) event, you can change the text in the column headers. In that event, you can use the `e-header-cell` class to identify the header cell element and update its text value.

The following code example shows how to change the text in the column headers.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").BeforeCellRender("beforeCellRender").
Render()
<script>
 function beforeCellRender(args) {
 // Condition to check whether the rendered element is header cell.
 if (
 args.colIndex >= 0 &&
 args.colIndex <= 10 &&
 args.element.classList.contains('e-header-cell')
) {
 var text = 'custom header ' + args.colIndex.toString();
 // Add the custom text to the innerText of the element.
 args.element.innerText = text;
 }
 }
</script>

```

### Limitations of insert and delete

The following features have some limitations in Insert/Delete:

- Insert row/column between the formatting applied cells.
- Insert row/column between the data validation.
- Insert row/column between the conditional formatting applied cells.
- Insert/delete row/column between the filter applied cells.

### See Also

- [Hyperlink](#)
- [Sorting](#)
- [Filtering](#)

### Undo and Redo in Spreadsheet control

**Undo** option helps you to undone the last action performed and **Redo** option helps you to do the same action which is reverted in the Spreadsheet. You can use the [allowUndoRedo](#) property to enable or disable undo redo functionality in spreadsheet.

**Note:** \* The default value for `allowUndoRedo` property is `true`.

By default, the `UndoRedo` module is injected internally into Spreadsheet to perform undo redo.

### Undo

It reverses the last action you performed with Spreadsheet. Undo can be done by any of the following ways:

- Select the undo item from HOME tab in Ribbon toolbar.
- Use `Ctrl + Z` keyboard shortcut to perform the undo.
- Use the `undo` method programmatically.

### Redo

It reverses the last undo action you performed with Spreadsheet. Redo can be done by any of the following ways:

- Select the redo item from HOME tab in Ribbon toolbar.
- Use `Ctrl + Y` keyboard shortcut to perform the redo.
- Use the `redo` method programmatically.

### Update custom actions in UndoRedo collection

You can update your own custom actions in `UndoRedo` collection, by using the `updateUndoRedoCollection` method. And also customize the undo redo operations of your custom action by using `actionComplete` event.

The following code example shows `How to update and customize your own actions for undo redo` functionality in the Spreadsheet control.

### CSHTML

```
<button id="customBtn" class="e-btn"> add/remove Class</button>
@Html.EJS().Spreadsheet("spreadsheet").AllowUndoRedo(true).ActionComplete("onActionComplete").Render()
<script>
 document.getElementById("customBtn").addEventListener('click',
updateCollection);
 function onActionComplete(args) {
 var actionEvents = args;
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (actionEvents.eventArgs.action == "customCSS") {
 var Element =
spreadsheetObj.getCell(actionEvents.eventArgs.rowIdx,
actionEvents.eventArgs.colIdx);
 if (actionEvents.eventArgs.requestType == "undo") {
 Element.classList.remove('customClass'); // To remove the
custom class in undo action
 }
 else {
 Element.classList.add('customClass'); // To add the custom
class in redo action
 }
 }
 }
}
```

```

 }
 function updateCollection() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet'),
 cell = spreadsheetObj.getActiveSheet().activeCell,
 cellIdx = ej.spreadsheet.getRangeIndexes(cell),
 Element = spreadsheetObj.getCell(cellIdx[0], cellIdx[1]);
 if (!Element.classList.contains("customClass")) {
 Element.classList.add('customClass'); // To add the custom class
in active cell element
 spreadsheetObj.updateUndoRedoCollection({ eventArgs: { class:
'customClass', rowIdx: cellIdx[0], colIdx: cellIdx[1], action: 'customCSS' }
}); // To update the undo redo collection
 }
 }
</script>
<style>
 .customClass.e-cell {
 background-color: red;
 }
</style>

```

### UNDOREDOCONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

See Also

- [Sorting](#)
- [Filtering](#)
- [Hyperlink](#)

### Find and Replace in Spreadsheet control

Find and Replace helps you to search for the target text and replace the found text with alternative text within the sheet or workbook. You can use the [allowFindAndReplace](#) property to enable or disable the Find and Replace functionality.

**Note:** \* The default value for `allowFindAndReplace` property is `true`.

#### Find

Find feature is used to select the matched contents of a cell within the sheet or workbook. It is extremely useful when working with large set of data source.

#### User Interface:

Find can be done by any of the following ways:

- Select the Search icon in the Ribbon toolbar or use `Ctrl + F` key to open the Find dialog.
- Use find Next and find Previous buttons to search the given value in the workbook.

- Select the option button in Find dialog to open the Find and Replace dialog. Then, select the below properties for enhanced searching.

**Note:** \* **Search within:** To search the target in a sheet (default) or in an entire workbook.

<br/> \* **Search by:** It enhance the search, either By Rows (default), or By Columns.

<br/> \* **Match case:** To find the matched value with case sensitive.

<br/> \* **Match exact cell contents:** To find the exact matched cell value with entire match cases.

- Using `find()` method to perform find operation.

## Replace

Replace feature is used to change the find contents of a cell within sheet or workbook. Replace All is used to change all the matched contents of a cell within sheet or workbook.

### User Interface:

Replace can be done by any of the following ways:

- Use `Ctrl + H` key to open the Find and Replace dialog.
- Use Replace button to change the found value in sheet or workbook.
- Using Replace All button, all the matched criteria can be replaced with find value based on sheet or workbook.
- Using `replace()` method to perform replace operation by passing the argument `args.replaceby` as `replace`.
- Using `replace()` method to perform replace all operation by passing the argument `args.replaceby` as `replaceall`.

## Go to

Go to feature is used to navigate to a specific cell address in the sheet or workbook.

### User Interface:

- Use `Ctrl + G` key to open the Go To dialog.
- Use `goTo()` method to perform Go To operation.

In the following sample, searching can be done by following ways:

- Select the Home tab in the Ribbon toolbar, and then select the Search icon.
- Enter any value in the search textbox.
- Select the next (or) previous button to find the entered value in the spreadsheet.
- You can have more options to find values by selecting the more options in the search toolbar.

## CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").AllowFindAndReplace(true).Sheets(sheet =>
{
```

```

sheet.Ranges(ranges =>
{
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();

}).Columns(column =>
{
 column.Width(110).Add();
 column.Width(220).Add();
 column.Width(90).Add();
 column.Width(140).Add();
 column.Width(90).Add();
 column.Width(100).Add();
 column.Width(100).Add();
}).Add();
}).Render()

```

### SEARCHCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 }
}

```

```

 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Limitations

- Undo/redo for Replace All is not supported in this feature.

## Accessibility in ASP.NET MVC Spreadsheet Control

The Spreadsheet control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Spreadsheet control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

```
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>
```

### WAI-ARIA attributes

The Spreadsheet control followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Spreadsheet control:

| Attributes                              | Purpose                                                                                                    |
|-----------------------------------------|------------------------------------------------------------------------------------------------------------|
| -----                                   | -----                                                                                                      |
| <code>grid</code> (role)                | This role is added to the spreadsheet content table and describes the collection of rows and columns.      |
| <code>gridcell</code> (role)            | This role is added to the cell element and describes the rows <code>&lt;td&gt;</code> element.             |
| <code>rowheader</code> (role)           | This role is added to the row header and describes the header of the rows.                                 |
| <code>colheader</code> (role)           | This role is added to the column header and describes the header of the columns.                           |
| <code>aria-rowindex</code> (attribute)  | This attribute describes the table's row index in the spreadsheet.                                         |
| <code>aria-colindex</code> (attribute)  | This attribute describes the table's column index in the spreadsheet.                                      |
| <code>aria-selected</code> (attribute)  | This attribute describes an item's (cell, menu, checkbox, etc.) current selected state in the spreadsheet. |
| <code>aria-rowcount</code> (attribute)  | This attribute describes the total number of rows in the table.                                            |
| <code>aria-colcount</code> (attribute)  | This attribute describes the total number of columns in the table.                                         |
| <code>aria-busy</code> (attribute)      | This attribute describes a currently updated or modified element.                                          |
| <code>aria-label</code> (attribute)     | This attribute describes the accessible name for the interactive elements.                                 |
| <code>textbox</code> (role)             | This role is assigned to the textbox that accepts text input.                                              |
| <code>menu</code> (role)                | This role has been added to the menu and describes the menu items.                                         |
| <code>aria-expanded</code> (attribute)  | This attribute describes the control (for example, dropdown) is expanded or collapsed.                     |
| <code>aria-multiline</code> (attribute) | This attribute defines what the Alt + Enter key does in the spreadsheet editor.                            |



### Keyboard interaction

The Spreadsheet control followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Spreadsheet control.

- | Press | To do this |
- |-----|-----|
- | Up arrow | Navigate from the active cell to the previous cell in the same column. |
- | Down arrow | Navigate from the active cell to the next cell in the same column. |
- | Left arrow | Navigate from the active cell to the previous cell in the same row. |
- | Right arrow | Navigate from the active cell to the next cell in the same row. |
- | Tab | Navigate the active cell to the next cell in the same row. |
- | Shift + Tab | Navigate the active cell to the previous cell in the same row. |
- | Home | Moves the selection to starting column in worksheet. |
- | Ctrl + Home | Move the selection to the first visible cell on a worksheet. |
- | Shift + Home | Extend the cell selection to the first column of a worksheet. |
- | Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |
- | Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |
- | Page Up | Move page up. |
- | Page Down | Move page down. |
- | Shift + Page Up | Perform page up by selecting all cells between. |
- | Shift + Page Down | Perform page down by selecting all cells between. |
- | Ctrl + Up | Navigate to the non-blank cell before the active cell in the same column. |
- | Ctrl + Down | Navigate to the last non-blank cell in the same column as the active cell. |
- | Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |
- | Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |
- | Shift + Up | Extend the selection of cells to the previous row. |
- | Shift + Down | Extend the selection of cells to the next row. |
- | Shift + Left | Extend the selection of cells to the previous column. |
- | Shift + Right | Extend the selection of cells to the next column. |
- | Ctrl + Shift + Up | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |
- | Ctrl + Shift + Down | Extend the cell selection to the last non-blank cell in the same column as the active cell. |
- | Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |

| Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |

| Enter | Navigate the active cell to the next cell in the same column. |

| Shift + Enter | Navigate to the previous cell in the same column from the active cell. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

| Shift + Enter | Complete the cell editing and select the cell above in the same column. |

| Tab | Complete the cell editing and select the next cell in the same row. |

| Shift + Tab | Complete the cell editing and select the previous cell in the same row. |

| Alt | Focus on the active ribbon tab. |

| Left | Move the focus to the previous items in the ribbon content. |

| Right | Move the focus to the next items in the ribbon content. |

| Alt + Down | Open the ribbon dropdown menu. |

| Esc / Alt + Up | Close the ribbon dropdown menu. |

### Ensuring accessibility

The Spreadsheet control's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Spreadsheet control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Spreadsheet control with accessibility tools.

### Keyboard Shortcuts And Navigation

The keyboard shortcuts supported in the spreadsheet are,

| Shortcut | Description |

|-----|-----|

| Ctrl + O | Displays dialog to open a file. |

| Ctrl + S / Alt + F2 | Saves the workbook. |

| F2 | Enables edit mode. |

| ESC | Cancel edit mode and discard the changes. |

| Backspace and SPACE | Clears content of the active cell and enables edit mode. |

| Ctrl + C | Copies the selected cells. |

| Ctrl + X | Cuts the selected cells. |

| Ctrl + V | Paste the clipboard(cut or copied) content in the new selected range. |

| Ctrl + B | Applies or removes **bold** formatting. |

| Ctrl + I | Applies or removes *italic* formatting. |

| Ctrl + U | Applies or removes underline. |

- | Ctrl + 5 | Applies or removes ~~strikethrough~~. |
- | Ctrl + Z | Reverses (Undo) the last action. |
- | Ctrl + Y | Repeats (Redo) the last reversed action. |
- | Ctrl + K | It opens the **Insert Hyperlink** dialog for adding new hyperlink to a cell. If the selected cell already contains hyperlink, it opens the **Edit Hyperlink** dialog. |
- | Ctrl + F / Shift + F5 | Opens **Find** dialog. |
- | Ctrl + H | Opens **Find and Replace** dialog. |
- | Ctrl + G | Opens **GoTo** dialog, which helps to navigate to cell. |
- | Ctrl + Shift + L | Applies filter to the first row of the selected range or used range. |
- | Alt + F | Opens the **File** menu. |
- | Alt + H | Go to **Home** tab. |
- | Alt + N | Go to **Insert** tab. |
- | Alt + M | Go to **Formulas** tab. |
- | Alt + A | Go to **Data** tab. |
- | Alt + W | Go to **View** tab. |
- | Tab | Navigate the active cell to the next cell in the same row. |
- | Shift + Tab | Navigate the active cell to the previous cell in the same row. |
- | Right or Left arrow | Move the focus to next or previous item in the tab if the focus is on ribbon tab. |
- | Up arrow | When a menu is open, move focus to the next item. |
- | Down arrow | When a menu is open, move focus to the previous item. |
- | Spacebar or Enter | Activate a selected button. |
- | Ctrl + F8 | Expand or collapse the ribbon content. |
- | Ctrl + Shift + U | Expand or collapse the formula bar. |
- | Ctrl + 9 | Hide the selected row. |
- | Ctrl + 0 | Hide the selected column. |
- | Home | Moves the selection to starting column in worksheet. |
- | Ctrl + Home | Move the selection to the first visible cell on a worksheet. |
- | Ctrl + Shift + Home | Extend the selection of cells to the beginning of the worksheet. |
- | Ctrl + End | Move to the last cell on a worksheet, right most last column and last row cell. |
- | Ctrl + & | Apply an outline border to the selected cells. |
- | Ctrl + Shift + & | Apply an outline border to the cells that you've chosen. |
- | Ctrl + Shift + ~ | Apply the **General** number format. |

- | Ctrl + Shift + \$ | Apply the **Currency** format with two decimal places (negative numbers in parentheses). |
- | Shift + F10 | Open the context menu. |
- | Ctrl + % | Apply the **Percentage** format with no decimal places. |
- | Ctrl + ^ | Apply the **Scientific** number format with two decimal places. |
- | Ctrl + Shift + # | Apply the **Date** format with the day, month, and year. |
- | Ctrl + Shift + @ | Apply the **Time** format with the hour and minute, and AM or PM. |
- | Ctrl + Shift + ! | Apply the **Number** format with two decimal places, thousands separator, and minus sign (-) for negative values. |
- | Ctrl + Spacebar | Select an entire column in a worksheet. |
- | Shift + Spacebar | Select an entire row in a worksheet. |
- | Shift + F3 | Opens **Insert Function** dialog. |
- | Ctrl + Alt + N | Opens new workbook. |
- | Shift + Page Down | Perform page down by selecting all cells between. |
- | Shift + Page Up | Perform page up by selecting all cells between. |
- | Ctrl + Left | Navigate to the non-blank cell before the active cell in the same row. |
- | Ctrl + Right | Navigate to the last non-blank cell in the same row as the active cell. |
- | Ctrl + Up | Navigate to the first non-blank cell in the same row as the active cell. |
- | Ctrl + Down | Navigate to the last cell that is not blank in the same column as the active cell. |
- | Ctrl + Shift + Left | Extend the cell selection to the previous non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Right | Extend the cell selection to the last non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Up | Extend the selection of cells to the first non-blank cell in the same row as the active cell. |
- | Ctrl + Shift + Down | Extend the selection of cells to the last non-blank cell in the same row as the active cell. |
- | Shift + Alt + K | Open the **List All Sheets** dropdown option. |
- | Up arrow | Navigate from the active cell to the previous cell in the same column. |
- | Down arrow | Navigate from the active cell to the next cell in the same column. |
- | Left arrow | Navigate from the active cell to the previous cell in the same row. |
- | Right arrow | Navigate from the active cell to the next cell in the same row. |
- | Page Up | Move page up. |
- | Page Down | Move page down. |
- | Shift + Up | Extend the selection of cells to the previous row. |

| Shift + Down | Extend the selection of cells to the next row. |

| Shift + Left | Extend the selection of cells to the previous column. |

| Shift + Right | Extend the selection of cells to the next column. |

| Ctrl + Top | Navigate to the non-blank cell before the active cell in the same column. |

| Ctrl + Shift + Top | Extend the cell selection to the previous non-blank cell in the same column as the active cell. |

| Ctrl + Shift + Bottom | Extend the cell selection to the last non-blank cell in the same column as the active cell. |

| Enter | Navigate the active cell to the next cell in the same column. |

| Shift + Enter | Navigate to the previous cell in the same column from the active cell. |

| Alt + Down | Open the list data validation menu and filter menu. |

| Alt + Up | Close the list data validation menu and filter menu. |

| Delete | Remove the contents of the cell. |

| Shift + Home | Extend the cell selection to the first column of a worksheet. |

| Shift + F11 | Add a new sheet. |

| Ctrl + Shift + 9 | Unhide the selected rows. |

| Ctrl + Shift + 0 | Unhide the selected columns. |

| Ctrl + D | Fill the cell down. |

| Ctrl + R | Fill the cell right. |

| Alt + Enter | While editing, add a new line. |

| Enter | Complete the cell editing and select the cell below in the same column. |

| Shift + Enter | Complete the cell editing and select the cell above in the same column. |

| Tab | Complete the cell editing and select the next cell in the same row. |

| Shift + Tab | Complete the cell editing and select the previous cell in the same row. |

| Alt | Focus on the active ribbon tab. |

| Left | Move the focus to the previous items in the ribbon content. |

| Right | Move the focus to the next items in the ribbon content. |

| Alt + Down | Open the ribbon dropdown menu. |

| Esc / Alt + Up | Close the ribbon dropdown menu. |

[See Also](#)

- [Formatting](#)

## Ribbon in Spreadsheet control

It helps to organize a spreadsheet's features into a series of tabs. By clicking the expand or collapse icon, you can expand or collapse the ribbon toolbar dynamically.

### Ribbon Customization

You can customize the ribbon using the following methods,

- | Method                           | Action                                                  |
|----------------------------------|---------------------------------------------------------|
| <code>hideRibbonTabs</code>      | Used to show or hide the existing ribbon tabs.          |
| <code>enableRibbonTabs</code>    | Used to enable or disable the existing ribbon tabs.     |
| <code>addRibbonTabs</code>       | Used to add custom ribbon tabs.                         |
| <code>hideToolbarItems</code>    | Used to show or hide the existing ribbon toolbar items. |
| <code>enableToolbarItems</code>  | Used to enable or disable the specified toolbar items.  |
| <code>addToolbarItems</code>     | Used to add the custom items in ribbon toolbar.         |
| <code>hideFileMenuItems</code>   | Used to show or hide the ribbon file menu items.        |
| <code>enableFileMenuItems</code> | Used to enable or disable file menu items.              |
| <code>addFileMenuItems</code>    | Used to add custom file menu items.                     |

The following code example shows the usage of ribbon customization.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Created("created").FileMenuBeforeOpen(
 "fileMenuBeforeOpen").FileMenuItemSelect("fileMenuItemSelect").Sheets(sheet
=>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.defaultData).StartCell("A1").
 Add();
 }).Columns(column =>
 {
 column.Width(180).Add();
 column.Width(130).Add();
 column.Width(130).Add();
 column.Width(180).Add();
 column.Width(130).Add();
 column.Width(120).Add();
 })
})
```

```

 }).Add();
 }).Render()
</script>

function created() {
 this.cellFormat({ fontWeight: 'bold', textAlign: 'center' },
'A1:F1');
 // Hiding the `Insert` from ribbon.
 this.hideRibbonTabs(['Insert']);
 // Set disabled state to `View` ribbon tab.
 this.enableRibbonTabs(['View'], false);
 // Adding the `Help` ribbon tab at the last index.
 // Specify the ribbon tab header text in last optional
argument(`insertBefore`) for inserting new tab before any existing tab.
 this.addRibbonTabs([header: { text: 'Help' }, content: [{ text:
'Feedback', tooltipText: 'Feedback',
 click: (args) => { /* Any click action for this toolbar item
will come here. */ } }]]);
 // Hiding the unwanted toolbar items from `Home` by specifying its
index.
 this.hideToolbarItems('Home', [0, 1, 2, 4, 14, 15, 21, 24]);
 // Set disable state to `Underline`, `Wrap text` toolbar items from
`Home` by specifying the item id.
 this.enableToolbarItems('Home', [`_${this.element.id}_underline`,
`_${this.element.id}_wrap`], false);
 // Set disable state to `Protect Sheet` toolbar item from `Data` by
mentioning its index.
 this.enableToolbarItems('Data', [0], false);
 // Adding the new `Custom Formulas` toolbar item under `Formulas`
tab for adding custom formulas.
 this.addToolbarItems(
 'Formulas', [{ type: 'Separator' }, {
 text: 'Custom Formulas', tooltipText: 'Custom Formulas',
 // You can set click handler for each new custom toolbar
item
 click: (args) => {
 // You can add custom formulas here.
 }
 }
]);
 // Adding new custom item `Import` after the existing `Open` item.
By default, new item will add after the specified item.
 this.addFileMenuItems([{ text: 'Import', iconCss: 'e-open e-icons'
}], 'Open');
 // Adding new custom item `Export As` after the existing `Save As`
item.
 // Set `insertAfter` optional argument as `false` for adding new
item before the specified item.
 this.addFileMenuItems(
 [
 {
 text: 'Export As', iconCss: 'e-save e-icons', items: [
text: 'XLSX', iconCss: 'e-xlsx e-icons' },
 { text: 'XLS', iconCss: 'e-xls e-icons' }, { text:
'CSV', iconCss: 'e-csv e-icons' }
],
 },
 'Save As', false);
]
)
}

```

```

function fileMenuBeforeOpen() { // Because the file menu items are
 created dynamically, you need to perform the hide or show and enable/disable
 operations
 // under filemenu before open event.
 // Hiding the `Save As` and `Open` item.
 this.hideFileMenuItems(['Save As', 'Open']);
 // Set disable state to `New` item. You can perform any file menu
 items customization by specifying the item id,
 // if it has more than one same item text. Set the last argument
 `isUniqueId` as true for using the item id.
 this.enableFileMenuItems(['${this.element.id}_New'], false, true); }
function fileMenuItemSelect(args) {
 // Custom file menu items select handler
 switch (args.item.text) {
 case 'Import': select(`#${this.element.id}_fileUpload`,
this.element).click();
 break;
 case 'XLSX': this.save({ saveType: 'Xlsx' });
 break;
 case 'XLS': this.save({ saveType: 'Xls' });
 break;
 case 'CSV': this.save({ saveType: 'Csv' });
 break;
 }
}
</script>

```

### RIBBONCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 }
}

```



```

 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

See Also

- [Worksheet](#)
- [Rows and columns](#)

## FreezePanels in Spreadsheet control

Freeze Panels helps you to keep particular rows or columns visible when scrolling the sheet content in the spreadsheet. You can specify the number of frozen rows and columns using the `frozenRows` and `frozenColumns` properties inside the `Sheet` property.

### Apply freezepanes on UI

#### User Interface:

In the active spreadsheet, click the cell where you want to create freeze panes. Freeze panes can be done in any of the following ways:

- Select the View tab in the Ribbon toolbar and choose the `Freeze Panes` item.
- Use the `freezePanels` method programmatically.

### FrozenRows

It allows you to keep a certain number of rows visible while scrolling vertically through the rest of the worksheet.

#### User Interface:

In the active spreadsheet, select the cell where you want to create frozen rows. Frozen rows can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the `Freeze Rows` item.

- You can specify the number of frozen rows using the `frozenRows` property inside the `Sheet` property.

### FrozenColumns

It allows you to keep a certain number of columns visible while scrolling horizontally through the rest of the worksheet.

#### User Interface:

In the active spreadsheet, select the cell where you want to create frozen columns. Frozen columns can be done in any one of the following ways:

- Select the View tab in the Ribbon toolbar and choose the `Freeze Columns` item.
- You can specify the number of frozen columns using the `frozenColumns` property inside the `Sheet` property.

In this demo, the `frozenColumns` is set as '2', and the `frozenRows` is set as '2'. Hence, the two columns on the left and the top two rows are frozen.

#### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").Created("createdHandler").Sheets(sheet =>
{
 sheet.Name("Gross Salary").FrozenRows(2).FrozenColumns(2).SelectedRange("C1").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).StartCell("A2").Add();
 }).Rows(row =>{
 row.Cells(cell => {
 cell.Index(1).Value("Period").Style(new SpreadsheetCellStyle { FontSize = "12pt", FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center, VerticalAlign = VerticalAlign.Middle }).Add();
 cell.Index(3).Value("Total Gross Salary").Style(new SpreadsheetCellStyle { FontSize = "12pt", FontWeight = FontWeight.Bold, TextAlign = TextAlign.Center, VerticalAlign = VerticalAlign.Middle }).Add();
 }).Add();
 row.Cells(cell =>
 {
 cell.Index(2).Value("Basic Salary").Add();
 cell.Value("Revised Basic Salary").Add();
 cell.Value("DA").Add();
 cell.Value("Revised DA").Add();
 cell.Value("HRA").Add();
 cell.Value("Revised HRA").Add();
 cell.Value("Conveyance Allowance").Add();
 cell.Value("Revised Conveyance Allowance").Add();
 cell.Value("Medical Expenses").Add();
 cell.Value("Revised Medical Expenses").Add();
 cell.Value("Special Allowance").Add();
 cell.Value("Revised Spcial Allowance").Add();
 cell.Value("Total Gross Salary").Add();
 cell.Value("Revised Total Gross Salary").Add();
 }
 }
}
```

```

 }).Add();
 row.Index(26).Cells(cell => {
 cell.Index(13).Value("Total Amount").Style(new
 SpreadsheetCellStyle { FontSize = "12pt", FontWeight = FontWeight.Bold,
 TextAlign = TextAlign.Center, VerticalAlign = VerticalAlign.Middle }).Add();
 cell.Formula("=SUM(O4:O26)").Style(new SpreadsheetCellStyle {
 FontSize = "12pt", FontWeight = FontWeight.Bold, TextAlign =
 TextAlign.Center, VerticalAlign = VerticalAlign.Middle }).Add();
 cell.Formula("=SUM(P4:P26)").Style(new SpreadsheetCellStyle {
 FontSize = "12pt", FontWeight = FontWeight.Bold, TextAlign =
 TextAlign.Center, VerticalAlign = VerticalAlign.Middle }).Add();
 }).Add();
 }).Columns(column => {
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function createdHandler() {
 this.wrap("C2:P2");
 this.merge('A1:B1');
 this.merge('C1:P1');
 this.cellFormat({ backgroundColor: '#4e4ee6', color: '#FFFFFF4',
 fontSize: '12pt', fontWeight: 'bold', textAlign: 'center', verticalAlign:
 'middle' }, 'A1:P2');
 this.cellFormat({ backgroundColor: '#4e4ee6', color: '#FFFFFF4' },
 'A3:B26');
 this.numberFormat('$#,##0.00', 'C2:P26');
 this.numberFormat('$#,##0.00', 'O27:P27');
 }
</script>

```

### FREEZEPANE.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Month= "January", Year= "2019", BasicSalary= 15100,
 RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
 RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
 1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=

```

```

1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C3,E3,G3,I3,K3,M3)", RevisedTotalGrossSalary=
"=SUM(D3,F3,H3,J3,L3,N3)", },
 new { Month= "February", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C4,E4,G4,I4,K4,M4)", RevisedTotalGrossSalary=
"=SUM(D4,F4,H4,J4,L4,N4)", },
 new { Month= "March", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C5,E5,G5,I5,K5,M5)", RevisedTotalGrossSalary=
"=SUM(D5,F5,H5,J5,L5,N5)", },
 new { Month= "April", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C6,E6,G6,I6,K6,M6)", RevisedTotalGrossSalary=
"=SUM(D6,F6,H6,J6,L6,N6)", },
 new { Month= "May", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C7,E7,G7,I7,K7,M7)", RevisedTotalGrossSalary=
"=SUM(D7,F7,H7,J7,L7,N7)", },
 new { Month= "June", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C8,E8,G8,I8,K8,M8)", RevisedTotalGrossSalary=
"=SUM(D8,F8,H8,J8,L8,N8)", },
 new { Month= "July", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C9,E9,G9,I9,K9,M9)", RevisedTotalGrossSalary=
"=SUM(D9,F9,H9,J9,L9,N9)", },
 new { Month= "August", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C10,E10,G10,I10,K10,M10)", RevisedTotalGrossSalary=
"=SUM(D10,F10,H10,J10,L10,N10)", },
 new { Month= "September", Year= "2019", BasicSalary=
15100, RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=

```

```

"=SUM(C11,E11,G11,I11,K11,M11)", RevisedTotalGrossSalary=
"=SUM(D11,F11,H11,J11,L11,N11)", },
 new { Month= "October", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C12,E12,G12,I12,K12,M12)", RevisedTotalGrossSalary=
"=SUM(D12,F12,H12,J12,L12,N12)", },
 new { Month= "November", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C13,E13,G13,I13,K13,M13)", RevisedTotalGrossSalary=
"=SUM(D13,F13,H13,J13,L13,N13)", },
 new { Month= "December", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C14,E14,G14,I14,K14,M14)", RevisedTotalGrossSalary=
"=SUM(D14,F14,H14,J14,L14,N14)", },
 new { Month= "January", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C15,E15,G15,I15,K15,M15)", RevisedTotalGrossSalary=
"=SUM(D15,F15,H15,J15,L15,N15)", },
 new { Month= "February", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C16,E16,G16,I16,K16,M16)", RevisedTotalGrossSalary=
"=SUM(D16,F16,H16,J16,L16,N16)", },
 new { Month= "March", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C17,E17,G17,I17,K17,M17)", RevisedTotalGrossSalary=
"=SUM(D17,F17,H17,J17,L17,N17)", },
 new { Month= "April", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C18,E18,G18,I18,K18,M18)", RevisedTotalGrossSalary=
"=SUM(D18,F18,H18,J18,L18,N18)", },
 new { Month= "May", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=

```

```

"=SUM(C19,E19,G19,I19,K19,M19)", RevisedTotalGrossSalary=
"=SUM(D19,F19,H19,J19,L19,N19)", },
 new { Month= "June", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C20,E20,G20,I20,K20,M20)", RevisedTotalGrossSalary=
"=SUM(D20,F20,H20,J20,L20,N20)", },
 new { Month= "July", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C21,E21,G21,I21,K21,M21)", RevisedTotalGrossSalary=
"=SUM(D21,F21,H21,J21,L21,N21)", },
 new { Month= "August", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C22,E22,G22,I22,K22,M22)", RevisedTotalGrossSalary=
"=SUM(D22,F22,H22,J22,L22,N22)", },
 new { Month= "September", Year= "2020", BasicSalary=
16610, RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C23,E23,G23,I23,K23,M23)", RevisedTotalGrossSalary=
"=SUM(D23,F23,H23,J23,L23,N23)", },
 new { Month= "October", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C24,E24,G24,I24,K24,M24)", RevisedTotalGrossSalary=
"=SUM(D24,F24,H24,J24,L24,N24)", },
 new { Month= "November", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C25,E25,G25,I25,K25,M25)", RevisedTotalGrossSalary=
"=SUM(D25,F25,H25,J25,L25,N25)", },
 new { Month= "December", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C26,E26,G26,I26,K26,M26)", RevisedTotalGrossSalary=
"=SUM(D26,F26,H26,J26,L26,N26)", }
};
ViewBag.DefaultData = data;
return View();
}

```

### Limitations

Freeze Panes feature is not compatible with all the features which are available in the spreadsheet. Below features are not compatible with Freeze Panes feature.

- Merging the cells between freeze and unfreeze area.
- If images and charts are added inside the freeze area cells, their portion in the unfreeze area will not move when scrolling.

### See Also

- [Sorting](#)
- [Filtering](#)
- [Undo Redo](#)

### Context Menu in Spreadsheet control

Context Menu is used to improve user interaction with Spreadsheet using the popup menu. This will open when right-clicking on Cell/Column Header/Row Header/ Pager in the Spreadsheet. You can use [enableContextMenu](#) property to enable/disable context menu.

**Note:** The default value for the `enableContextMenu` property is `true`.

#### Context Menu Items in Row Cell

Find the table below for default context menu items and their actions.

| Context Menu items   | Action                                                                                                                                      |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| ----- -----          |                                                                                                                                             |
| <b>Cut</b>           | Cut the selected cells data to the clipboard, you can select a cell where you want to move the data.                                        |
| <b>Copy</b>          | Copy the selected cells data to the clipboard, so that you can paste it to somewhere else.                                                  |
| <b>Paste</b>         | Paste the data from clipboard to spreadsheet.                                                                                               |
| <b>Paste Special</b> | <b>Values</b> - Paste the data values from clipboard to spreadsheet. <b>Formats</b> - Paste the data formats from clipboard to spreadsheet. |
| <b>Filter</b>        | Perform filtering to the selected cells based on an active cell's value.                                                                    |
| <b>Sort</b>          | Perform sorting to the selected range of cells by ascending or descending.                                                                  |
| <b>Hyperlink</b>     | Create a link in the spreadsheet to navigate to web links or cell reference within the sheet or other sheets in the Spreadsheet.            |

#### Context Menu Items in Row Header / Column Header

Find the table below for default context menu items and their actions.

| Context Menu items | Action                                                                                                           |
|--------------------|------------------------------------------------------------------------------------------------------------------|
| ----- -----        |                                                                                                                  |
| <b>Cut</b>         | Cut the selected row/column header data to the clipboard, you can select a cell where you want to move the data. |

| **Copy** | Copy the selected row/column header data to the clipboard, so that you can paste it to somewhere else. |

| **Paste** | Paste the data from clipboard to spreadsheet. |

| **Paste Special** | **Values** - Paste the data values from clipboard to spreadsheet. **Formats** - Paste the data formats from clipboard to spreadsheet. |

| **Insert Columns** | Insert new rows or columns into the worksheet. |

| **Delete Columns** | Delete existing rows or columns from the worksheet. |

| **Hide Columns** | Hide the rows and columns. |

| **UnHide Columns** | Show the hidden rows and columns. |

### Context Menu Items in Pager

Find the table below for default context menu items and their actions.

| Context Menu items | Action |

|-----|-----|

| **Insert** | Insert a new worksheet in front of an existing worksheet in the spreadsheet. |

| **Delete** | Delete the selected worksheet from the spreadsheet. |

| **Rename** | Rename the selected worksheet. |

| **Protect Sheet** | Prevent unwanted changes from others by limiting their ability to edit. |

| **Hide** | Hide the selected worksheet. |

### Context Menu Customization

You can perform the following context menu customization options in the spreadsheet

- Add Context Menu Items
- Remove Context Menu Items
- Enable/Disable Context Menu Items

### Add Context Menu Items

You can add the custom items in context menu using the **addContextMenuItems** in **contextmenuBeforeOpen** event

In this demo, Custom Item is added after the Paste item in the context menu.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").contextMenuBeforeOpen("createdHandler")
.Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>) ViewBag.DefaultData).Add();
 }).Columns(column =>
 {
 column.Width(80).Add();
 });
});
```



```

 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function createdHandler(args) {
 if (args.element.id === this.element.id + '_contextmenu') {
 this.addContextMenuItems([[text: 'Custom Item']], 'Paste Special',
false); //To pass the items, Item before / after that the element to be
inserted, Set false if the items need to be inserted before the text.
 }
 }
</script>
<style>
 .e-spreadsheet .e-tab .e-tab-text {
 display: inherit;
 }
</style>

```

### ADDCONTEXTMENU.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Month= "January", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C3,E3,G3,I3,K3,M3)", RevisedTotalGrossSalary=
"=SUM(D3,F3,H3,J3,L3,N3)", },
 new { Month= "February", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C4,E4,G4,I4,K4,M4)", RevisedTotalGrossSalary=
"=SUM(D4,F4,H4,J4,L4,N4)", },
 new { Month= "March", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=

```

```

1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C5,E5,G5,I5,K5,M5)", RevisedTotalGrossSalary=
"=SUM(D5,F5,H5,J5,L5,N5)", },
 new { Month= "April", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C6,E6,G6,I6,K6,M6)", RevisedTotalGrossSalary=
"=SUM(D6,F6,H6,J6,L6,N6)", },
 new { Month= "May", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C7,E7,G7,I7,K7,M7)", RevisedTotalGrossSalary=
"=SUM(D7,F7,H7,J7,L7,N7)", },
 new { Month= "June", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C8,E8,G8,I8,K8,M8)", RevisedTotalGrossSalary=
"=SUM(D8,F8,H8,J8,L8,N8)", },
 new { Month= "July", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C9,E9,G9,I9,K9,M9)", RevisedTotalGrossSalary=
"=SUM(D9,F9,H9,J9,L9,N9)", },
 new { Month= "August", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C10,E10,G10,I10,K10,M10)", RevisedTotalGrossSalary=
"=SUM(D10,F10,H10,J10,L10,N10)", },
 new { Month= "September", Year= "2019", BasicSalary=
15100, RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C11,E11,G11,I11,K11,M11)", RevisedTotalGrossSalary=
"=SUM(D11,F11,H11,J11,L11,N11)", },
 new { Month= "October", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C12,E12,G12,I12,K12,M12)", RevisedTotalGrossSalary=
"=SUM(D12,F12,H12,J12,L12,N12)", },
 new { Month= "November", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=

```

```

"=SUM(C13,E13,G13,I13,K13,M13)", RevisedTotalGrossSalary=
"=SUM(D13,F13,H13,J13,L13,N13)", },
 new { Month= "December", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C14,E14,G14,I14,K14,M14)", RevisedTotalGrossSalary=
"=SUM(D14,F14,H14,J14,L14,N14)", },
 new { Month= "January", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C15,E15,G15,I15,K15,M15)", RevisedTotalGrossSalary=
"=SUM(D15,F15,H15,J15,L15,N15)", },
 new { Month= "February", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C16,E16,G16,I16,K16,M16)", RevisedTotalGrossSalary=
"=SUM(D16,F16,H16,J16,L16,N16)", },
 new { Month= "March", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C17,E17,G17,I17,K17,M17)", RevisedTotalGrossSalary=
"=SUM(D17,F17,H17,J17,L17,N17)", },
 new { Month= "April", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C18,E18,G18,I18,K18,M18)", RevisedTotalGrossSalary=
"=SUM(D18,F18,H18,J18,L18,N18)", },
 new { Month= "May", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C19,E19,G19,I19,K19,M19)", RevisedTotalGrossSalary=
"=SUM(D19,F19,H19,J19,L19,N19)", },
 new { Month= "June", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C20,E20,G20,I20,K20,M20)", RevisedTotalGrossSalary=
"=SUM(D20,F20,H20,J20,L20,N20)", },
 new { Month= "July", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=

```

```

"=SUM(C21,E21,G21,I21,K21,M21)", RevisedTotalGrossSalary=
"=SUM(D21,F21,H21,J21,L21,N21)", },
 new { Month= "August", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C22,E22,G22,I22,K22,M22)", RevisedTotalGrossSalary=
"=SUM(D22,F22,H22,J22,L22,N22)", },
 new { Month= "September", Year= "2020", BasicSalary=
16610, RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C23,E23,G23,I23,K23,M23)", RevisedTotalGrossSalary=
"=SUM(D23,F23,H23,J23,L23,N23)", },
 new { Month= "October", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C24,E24,G24,I24,K24,M24)", RevisedTotalGrossSalary=
"=SUM(D24,F24,H24,J24,L24,N24)", },
 new { Month= "November", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C25,E25,G25,I25,K25,M25)", RevisedTotalGrossSalary=
"=SUM(D25,F25,H25,J25,L25,N25)", },
 new { Month= "December", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C26,E26,G26,I26,K26,M26)", RevisedTotalGrossSalary=
"=SUM(D26,F26,H26,J26,L26,N26)", }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Remove Context Menu Items

You can remove the items in context menu using the `removeContextMenuItems` in `contextmenuBeforeOpen` event

In this demo, Insert Column item has been removed from the row/column header context menu.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").contextMenuBeforeOpen("createdHandler")
.Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }
);
}
)

```

```

 }).Columns(column =>
 {
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function createdHandler() {
 // To remove existing context menu items.
 this.removeContextMenuItems(["Insert Column"], false);
 }
</script>
<style>
 .e-spreadsheet .e-tab .e-tab-text {
 display: inherit;
 }
</style>

```

### REMOVECONTEXTMENU.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Month= "January", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C3,E3,G3,I3,K3,M3)", RevisedTotalGrossSalary=
"=SUM(D3,F3,H3,J3,L3,N3)", },
 new { Month= "February", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C4,E4,G4,I4,K4,M4)", RevisedTotalGrossSalary=
"=SUM(D4,F4,H4,J4,L4,N4)", },
 new { Month= "March", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=

```

```

1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C5,E5,G5,I5,K5,M5)", RevisedTotalGrossSalary=
"=SUM(D5,F5,H5,J5,L5,N5)", },
 new { Month= "April", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C6,E6,G6,I6,K6,M6)", RevisedTotalGrossSalary=
"=SUM(D6,F6,H6,J6,L6,N6)", },
 new { Month= "May", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C7,E7,G7,I7,K7,M7)", RevisedTotalGrossSalary=
"=SUM(D7,F7,H7,J7,L7,N7)", },
 new { Month= "June", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C8,E8,G8,I8,K8,M8)", RevisedTotalGrossSalary=
"=SUM(D8,F8,H8,J8,L8,N8)", },
 new { Month= "July", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C9,E9,G9,I9,K9,M9)", RevisedTotalGrossSalary=
"=SUM(D9,F9,H9,J9,L9,N9)", },
 new { Month= "August", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C10,E10,G10,I10,K10,M10)", RevisedTotalGrossSalary=
"=SUM(D10,F10,H10,J10,L10,N10)", },
 new { Month= "September", Year= "2019", BasicSalary=
15100, RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C11,E11,G11,I11,K11,M11)", RevisedTotalGrossSalary=
"=SUM(D11,F11,H11,J11,L11,N11)", },
 new { Month= "October", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C12,E12,G12,I12,K12,M12)", RevisedTotalGrossSalary=
"=SUM(D12,F12,H12,J12,L12,N12)", },
 new { Month= "November", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=

```

```

1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C13,E13,G13,I13,K13,M13)", RevisedTotalGrossSalary=
"=SUM(D13,F13,H13,J13,L13,N13)", },
 new { Month= "December", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C14,E14,G14,I14,K14,M14)", RevisedTotalGrossSalary=
"=SUM(D14,F14,H14,J14,L14,N14)", },
 new { Month= "January", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C15,E15,G15,I15,K15,M15)", RevisedTotalGrossSalary=
"=SUM(D15,F15,H15,J15,L15,N15)", },
 new { Month= "February", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C16,E16,G16,I16,K16,M16)", RevisedTotalGrossSalary=
"=SUM(D16,F16,H16,J16,L16,N16)", },
 new { Month= "March", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C17,E17,G17,I17,K17,M17)", RevisedTotalGrossSalary=
"=SUM(D17,F17,H17,J17,L17,N17)", },
 new { Month= "April", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C18,E18,G18,I18,K18,M18)", RevisedTotalGrossSalary=
"=SUM(D18,F18,H18,J18,L18,N18)", },
 new { Month= "May", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C19,E19,G19,I19,K19,M19)", RevisedTotalGrossSalary=
"=SUM(D19,F19,H19,J19,L19,N19)", },
 new { Month= "June", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C20,E20,G20,I20,K20,M20)", RevisedTotalGrossSalary=
"=SUM(D20,F20,H20,J20,L20,N20)", },
 new { Month= "July", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=

```

```

"=SUM(C21,E21,G21,I21,K21,M21)", RevisedTotalGrossSalary=
"=SUM(D21,F21,H21,J21,L21,N21)", },
 new { Month= "August", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C22,E22,G22,I22,K22,M22)", RevisedTotalGrossSalary=
"=SUM(D22,F22,H22,J22,L22,N22)", },
 new { Month= "September", Year= "2020", BasicSalary=
16610, RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C23,E23,G23,I23,K23,M23)", RevisedTotalGrossSalary=
"=SUM(D23,F23,H23,J23,L23,N23)", },
 new { Month= "October", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C24,E24,G24,I24,K24,M24)", RevisedTotalGrossSalary=
"=SUM(D24,F24,H24,J24,L24,N24)", },
 new { Month= "November", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C25,E25,G25,I25,K25,M25)", RevisedTotalGrossSalary=
"=SUM(D25,F25,H25,J25,L25,N25)", },
 new { Month= "December", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C26,E26,G26,I26,K26,M26)", RevisedTotalGrossSalary=
"=SUM(D26,F26,H26,J26,L26,N26)", }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Enable/Disable Context Menu Items

You can enable/disable the items in context menu using the `enableContextMenuItems` in `contextmenuBeforeOpen` event

In this demo, Rename item is disabled in the pager context menu.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").contextMenuBeforeOpen("createdHandler")
.Sheets(sheet =>
{
 sheet.Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).Add();
 }
);
}
)

```



```

 }).Columns(column =>
 {
 column.Width(80).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(80).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 function createdHandler() {
 //To enable / disable context menu items.
 this.spreadsheet.enableContextMenuItems(['Rename'], false, false);
 // Contextmenu Items that needs to be enabled / disabled, Set true / false
 // to enable / disable the menu items, Set true if the given text is a unique
 // id.
 }
</script>
<style>
 .e-spreadsheet .e-tab .e-tab-text {
 display: inherit;
 }
</style>

```

### ENABLECONTEXTMENU.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { Month= "January", Year= "2019", BasicSalary= 15100,
 RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
 RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
 1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
 1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
 "=SUM(C3,E3,G3,I3,K3,M3)", RevisedTotalGrossSalary=
 "=SUM(D3,F3,H3,J3,L3,N3)", },
 new { Month= "February", Year= "2019", BasicSalary= 15100,
 RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
 RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
 1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
 1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
 "=SUM(C4,E4,G4,I4,K4,M4)", RevisedTotalGrossSalary=
 "=SUM(D4,F4,H4,J4,L4,N4)", },
 }
}

```

```

 new { Month= "March", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C5,E5,G5,I5,K5,M5)", RevisedTotalGrossSalary=
"=SUM(D5,F5,H5,J5,L5,N5)", },
 new { Month= "April", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C6,E6,G6,I6,K6,M6)", RevisedTotalGrossSalary=
"=SUM(D6,F6,H6,J6,L6,N6)", },
 new { Month= "May", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C7,E7,G7,I7,K7,M7)", RevisedTotalGrossSalary=
"=SUM(D7,F7,H7,J7,L7,N7)", },
 new { Month= "June", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C8,E8,G8,I8,K8,M8)", RevisedTotalGrossSalary=
"=SUM(D8,F8,H8,J8,L8,N8)", },
 new { Month= "July", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C9,E9,G9,I9,K9,M9)", RevisedTotalGrossSalary=
"=SUM(D9,F9,H9,J9,L9,N9)", },
 new { Month= "August", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C10,E10,G10,I10,K10,M10)", RevisedTotalGrossSalary=
"=SUM(D10,F10,H10,J10,L10,N10)", },
 new { Month= "September", Year= "2019", BasicSalary=
15100, RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C11,E11,G11,I11,K11,M11)", RevisedTotalGrossSalary=
"=SUM(D11,F11,H11,J11,L11,N11)", },
 new { Month= "October", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C12,E12,G12,I12,K12,M12)", RevisedTotalGrossSalary=
"=SUM(D12,F12,H12,J12,L12,N12)", },

```

```

 new { Month= "November", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C13,E13,G13,I13,K13,M13)", RevisedTotalGrossSalary=
"=SUM(D13,F13,H13,J13,L13,N13)", },
 new { Month= "December", Year= "2019", BasicSalary= 15100,
RevisedBasicSalary= 15800, DA= 2000, RevisedDA= 2200, HRA= 5000,
RevisedHRA= 6500, ConveyanceAllowance= 1500, RevisedConveyanceAllowance=
1700, MedicalExpenses= 1250, RevisedMedicalExpenses= 1500, SpecialAllowance=
1000, RevisedSpecialAllowance= 1200, TotalGrossSalary=
"=SUM(C14,E14,G14,I14,K14,M14)", RevisedTotalGrossSalary=
"=SUM(D14,F14,H14,J14,L14,N14)", },
 new { Month= "January", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C15,E15,G15,I15,K15,M15)", RevisedTotalGrossSalary=
"=SUM(D15,F15,H15,J15,L15,N15)", },
 new { Month= "February", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C16,E16,G16,I16,K16,M16)", RevisedTotalGrossSalary=
"=SUM(D16,F16,H16,J16,L16,N16)", },
 new { Month= "March", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C17,E17,G17,I17,K17,M17)", RevisedTotalGrossSalary=
"=SUM(D17,F17,H17,J17,L17,N17)", },
 new { Month= "April", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C18,E18,G18,I18,K18,M18)", RevisedTotalGrossSalary=
"=SUM(D18,F18,H18,J18,L18,N18)", },
 new { Month= "May", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C19,E19,G19,I19,K19,M19)", RevisedTotalGrossSalary=
"=SUM(D19,F19,H19,J19,L19,N19)", },
 new { Month= "June", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C20,E20,G20,I20,K20,M20)", RevisedTotalGrossSalary=
"=SUM(D20,F20,H20,J20,L20,N20)", },

```

```

 new { Month= "July", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C21,E21,G21,I21,K21,M21)", RevisedTotalGrossSalary=
"=SUM(D21,F21,H21,J21,L21,N21)", },
 new { Month= "August", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C22,E22,G22,I22,K22,M22)", RevisedTotalGrossSalary=
"=SUM(D22,F22,H22,J22,L22,N22)", },
 new { Month= "September", Year= "2020", BasicSalary=
16610, RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C23,E23,G23,I23,K23,M23)", RevisedTotalGrossSalary=
"=SUM(D23,F23,H23,J23,L23,N23)", },
 new { Month= "October", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C24,E24,G24,I24,K24,M24)", RevisedTotalGrossSalary=
"=SUM(D24,F24,H24,J24,L24,N24)", },
 new { Month= "November", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C25,E25,G25,I25,K25,M25)", RevisedTotalGrossSalary=
"=SUM(D25,F25,H25,J25,L25,N25)", },
 new { Month= "December", Year= "2020", BasicSalary= 16610,
RevisedBasicSalary= 17380, DA= 2200, RevisedDA= 2420, HRA= 5500,
RevisedHRA= 7150, ConveyanceAllowance= 1650, RevisedConveyanceAllowance=
1870, MedicalExpenses= 1375, RevisedMedicalExpenses= 1650, SpecialAllowance=
1100, RevisedSpecialAllowance= 1320, TotalGrossSalary=
"=SUM(C26,E26,G26,I26,K26,M26)", RevisedTotalGrossSalary=
"=SUM(D26,F26,H26,J26,L26,N26)", }
 };
 ViewBag.DefaultData = data;
 return View();
}

```

See Also

- [Worksheet](#)
- [Rows and columns](#)

## Cell Template in Spreadsheet Control

Cell Template is used for adding HTML elements into Spreadsheet. You can add the cell template in spreadsheet by using the `template` property and specify the address using the `address` property inside the `ranges` property. You can customize the HTML elements similar to Syncfusion components (TextBox, DropDownList, RadioButton, MultiSelect, DatePicker etc) by using the `beforeCellRender` event. In this demo, Cell template is applied to `C2:C9` and instantiated with HTML input components like TextBox, RadioButton, TextArea. You need to bind the events to perform any operations through HTML elements or Syncfusion components. Here, we have added `change` event in to the MultiSelect control, and we have updated the selected data into the spreadsheet cell through that change event.

The following code example describes the above behavior.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ShowRibbon(false).ShowFormulaBar(false).AllowEditing(false).Created("createdHandler").BeforeSelect("beforeSelectHandler").BeforeCellRender("beforeCellRenderHandler").ScrollSettings(scrollSettings =>
{
 scrollSettings.EnableVirtualization(false).IsFinite(true);
}).Sheets(sheet =>
{
 sheet.Name("Registration Form").RowCount(40).ColCount(30).ShowGridLines(false).Rows(row =>
 {
 row.Height(55).Cells(cell => cell.Index(1).Value("Interview Registration Form").Style(style =>
 {
 style.FontWeight(FontWeight.Bold).TextAlign(TextAlign.Center).VerticalAlign(VERTICAL_ALIGN_MIDDLE).FontSize("12pt");
 }).Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Name").Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Date of Birth:").Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Gender:").Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Year of Experience:").Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Areas of Interest:").Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Mobile Number:").Add()).Add();
 row.Height(45).Cells(cell => cell.Index(1).Value("Email:").Add()).Add();
 row.Height(82).Cells(cell => cell.Index(1).Value("Address:").Add()).Add();
 }).Columns(column =>
 {
 column.Index(1).Width(190).Add();
 column.Width(350).Add();
 }).Ranges(ranges =>
 {
 ranges.Template("<input />").Address("C2:C3").Add();
 });
});
```

```

 ranges.Template("<div><input type='radio' name='gender'
value='male' /><input type='radio' name='gender'
value='female' /></div>").Address("C4").Add();
 ranges.Template("<input />").Address("C5:C8").Add();
 ranges.Template("<textarea rows='3' />").Address("C9").Add();
 ranges.Template("<button class='e-btn e-flat'
style='float:right'>Add</button>").Address("C11").Add();
 }).Add();
}).Render()
<script>
function createdHandler() {
 //Applies format to specified range
 this.cellFormat({ fontWeight: 'bold' }, 'B2:B9');
}
function beforeSelectHandler(args) {
 //Prevents selection
 args.cancel = true;
}
function beforeCellRenderHandler(args) {
 //Initializing input components before cell rendering
 if (this.activeSheetIndex === 0) {
 var target = args.element.firstElementChild;
 switch (args.address) {
 case 'B1':
 args.element.colSpan = 2;
 break;
 case 'C2':
 new ej.inputs.TextBox({ placeholder: 'Name' }, target);
 break;
 case 'C3':
 new ej.calendars.DatePicker({ placeholder: 'DOB', },
target);
 break;
 case 'C4':
 new ej.buttons.RadioButton({ label: 'Male' },
args.element.firstElementChild.firstElementChild);
 new ej.buttons.RadioButton({ label: 'Female' },
args.element.firstElementChild.lastElementChild);
 break;
 case 'C5':
 var experience = ['0 - 1 year', '1 - 3 years', '3 - 5
years', '5 - 10 years'];
 new ej.dropdowns.DropDownList({
 placeholder: 'Experience',
 dataSource: experience
 }, target);
 break;
 case 'C6':
 var languages = ['JAVA', 'C#', 'SQL'];
 new ej.dropdowns.MultiSelect({
 showClearButton: false,
 placeholder: 'Areas of Interest',
 dataSource: languages,
 change: function(evt) {
 if (args.cell) {
 debugger
 args.cell.value = evt.value.toString();

```

```

 } else {
 var range =
ej.spreadsheet.getRangeIndexes(evt.address);

spreadsheet.sheets[spreadsheet.activeSheetIndex].rows[range[0]].cells[range[
1]] = { value: evt.value.toString() };
 }
 }
 }, target);
 break;
 case 'C7':
 new ej.inputs.TextBox({
 placeholder: 'Mobile Number'
 }, target);
 break;
 case 'C8':
 new ej.inputs.TextBox({
 placeholder: 'Email'
 }, target);
 break;
 case 'C9':
 new ej.inputs.TextBox(null, target);
 break;
 }
}
}
</script>
<style>
.e-spreadsheet .e-tab .e-tab-text {
 display: inherit;
}
.e-spreadsheet .e-cell .e-radio-wrapper {
 margin: 5px;
}
.e-spreadsheet .e-cell .e-radio-wrapper:first-child {
 margin-left: 0;
}
.e-spreadsheet .e-cell .e-radio + label .e-label {
 color: rgba(0, 0, 0, 0.87);
}
</style>

```

**TEMPLATE.CS**

```

public IActionResult Index()
{
 return View();
}

```

See Also

- [Worksheet](#)
- [Rows and columns](#)

## Globalization in ASP.NET MVC Spreadsheet control

### Localization

The [Localization](#) library allows you to localize the default text content of the Spreadsheet. The Spreadsheet has static text on some features (cell formatting, Merge, Data validation, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [locale](#) value and translation object.

The following list of properties and their values are used in the Spreadsheet.

|Locale keywords|Text|

|-----|----|

Cut | Cut

Copy | Copy

Paste | Paste

PasteSpecial | Paste Special

All | All

Values | Values

Formats | Formats

Font | Font

FontSize | Font Size

Bold | Bold

Italic | Italic

Underline | Underline

Strikethrough | Strikethrough

TextColor | Text Color

FillColor | Fill Color

HorizontalAlignment | Horizontal Alignment

AlignLeft | Align Left

AlignCenter | Center

AlignRight | Align Right

VerticalAlignment | Vertical Alignment

AlignTop | Align Top

AlignMiddle | Align Middle

AlignBottom | Align Bottom

InsertFunction | Insert Function

Insert | Insert



Delete | Delete

Rename | Rename

Hide | Hide

Unhide | Unhide

NameBox | Name Box

ShowHeaders | Show Headers

HideHeaders | Hide Headers

ShowGridLines | Show Gridlines

HideGridLines | Hide Gridlines

AddSheet | Add Sheet

ListAllSheets | List All Sheets

FullScreen | Full Screen

CollapseToolbar | Collapse Toolbar

ExpandToolbar | Expand Toolbar

CollapseFormulaBar | Collapse Formula Bar

ExpandFormulaBar | Expand Formula Bar

File | File

Home | Home

Formulas | Formulas

View | View

New | New

Open | Open

SaveAs | Save As

ExcelXlsx | Microsoft Excel

ExcelXls | Microsoft Excel 97-2003

CSV | Comma-separated values

FormulaBar | Formula Bar

Ok | Ok

Close | Close

Cancel | Cancel

Apply | Apply

MoreColors | More Colors

StandardColors | Standard Colors

General | General

Number | Number

Currency | Currency

Accounting | Accounting

ShortDate | Short Date

LongDate | Long Date

Time | Time

Percentage | Percentage

Fraction | Fraction

Scientific | Scientific

Text | Text

NumberFormat | Number Format

MobileFormulaBarPlaceHolder | Enter value or Formula

PasteAlert | You can't paste it here because the copy area and paste area aren't in the same size. Try pasting in a different range.

DestroyAlert | Are you sure you want to destroy the current workbook without saving and create a new workbook?

SheetRenameInvalidAlert | Sheet name contains invalid character.

SheetRenameEmptyAlert | Sheet name cannot be empty.

SheetRenameAlreadyExistsAlert | Sheet name already exists. Enter another name.

DeleteSheetAlert | Are you sure you want to delete this sheet?

DeleteSingleLastSheetAlert | A Workbook must contain at least one visible worksheet.

PickACategory | Pick a category

Description | Description

UnsupportedFile | Unsupported File

InvalidUrl | Invalid Url

SUM | Adds a series of numbers and/or cells.

SUMIF | Adds the cells based on the specified condition.

SUMIFS | Adds the cells based on the specified conditions.

ABS | Returns the value of a number without its sign.

RAND | Returns a random number between 0 and 1.

RANDBETWEEN | Returns a random integer based on the specified values.

FLOOR | Rounds a number down to the nearest multiple of a given factor.

CEILING | Rounds a number up to the nearest multiple of a given factor.

PRODUCT | Multiplies a series of numbers and/or cells.

AVERAGE | Calculates average for the series of numbers and/or cells excluding text.

AVERAGEIF | Calculates average for the cells based on the specified criterion.

AVERAGEIFS | Calculates average for the cells based on the specified conditions.

AVERAGEA | Calculates the average for the cells evaluating TRUE as 1 text and FALSE as 0.

COUNT | Counts the cells that contain numeric values in a range.

COUNTIF | Counts the cells based on the specified condition.

COUNTIFS | Counts the cells based on specified conditions.

COUNTA | Counts the cells that contain values in a range.

MIN | Returns the smallest number of the given arguments.

MAX | Returns the largest number of the given arguments.

DATE | Returns the date based on the given year, month, and day.

DAY | Returns the day from the given date.

DAYS | Returns the number of days between two dates.

IF | Returns value based on the given expression.

IFS | Returns value based on the given multiple expressions.

AND | Returns TRUE if all the arguments are TRUE otherwise returns FALSE.

OR | Returns TRUE if any of the arguments are TRUE otherwise returns FALSE.

IFERROR | Returns value if no error found else it will return specified value.

CHOOSE | Returns a value from the list of values based on the index number.

INDEX | Returns the value of the cell in a given range based on row and column number.

FIND | Returns the position of a string within another string which is case sensitive.

CONCATENATE | Combines two or more strings together.

CONCAT | Concatenates a list or a range of text strings.

SUBTOTAL | Returns subtotal for a range using the given function number.

RADIANS | Converts degrees into radians.

MATCH | Returns the relative position of a specified value in the given range.

DefineNameExists | This name already exists try a different name.

CircularReference | When a formula refers to one or more circular references this may result in an incorrect calculation.

ShowRowsWhere | Show rows where |

CustomFilterDatePlaceHolder | Choose a date

CustomFilterPlaceHolder | Enter the value

CustomFilter | Custom Filter

Between | Between

MatchCase | Match Case

DateTimeFilter | DateTime Filters

Undo | Undo

Redo | Redo

DateFilter | Date Filters

TextFilter | Text Filters

NumberFilter | Number Filters

ClearFilter | Clear Filter

NoResult | No Matches Found

FilterFalse | False

FilterTrue | True

Blanks | Blanks

SelectAll | Select All

GreaterThanOrEqual | Greater Than Or Equal

GreaterThan | Greater Than

LessThanOrEqual | Less Than Or Equal

LessThan | Less Than

NotEqual | Not Equal

Equal | Equal

Contains | Contains

EndsWith | Ends With

StartsWith | Starts With

ClearButton | Clear

FilterButton | Filter

CancelButton | Cancel

OKButton | OK

Search | Search

Link | Link

Hyperlink | Hyperlink

EditHyperlink | Edit Hyperlink

OpenHyperlink | Open Hyperlink

RemoveHyperlink | Remove Hyperlink  
InsertLink | Insert Link  
EditLink | Edit Link  
WebPage | WEB PAGE  
ThisDocument | THIS DOCUMENT  
DisplayText | Display Text  
Url | URL  
CellReference | Cell Reference  
Sheet | Sheet  
DefinedNames | Defined Names  
EnterTheTextToDisplay | Enter the text to display  
EnterTheUrl | Enter the URL  
ProtectSheet | Protect Sheet  
UnprotectSheet | Unprotect Sheet  
SelectCells | Select cells  
FormatCells | Format cells  
FormatRows | Format rows  
Format Columns | Format columns  
InsertLinks | Insert links  
ProtectContent | Protect the contents of locked cells  
ProtectAllowUser | Allow all users of this worksheet to |  
EditAlert | The cell you're trying to change is protected. To make a change, unprotect the sheet.  
FindReplaceTooltip | Find & Replace  
InsertingEmptyValue | Reference value is not valid.  
ByRow | By Row  
ByColumn | By Column  
MatchExactCellElements | Match Exact Cell Contents  
EnterCellAddress | Enter Cell Address  
FindAndReplace | Find and Replace  
ReplaceAllEnd | matches replaced with.  
FindNextBtn | Find Next  
FindPreviousBtn | Find Previous  
ReplaceBtn | Replace

ReplaceAllBtn | Replace All  
GotoHeader | Go To  
GotoSpecialHeader | Go To Special  
SearchWithin | Search within  
SearchBy | Search by  
Reference | Reference  
Workbook | Workbook  
NoElements | We couldn't find what you were looking for.  
FindWhat | Find what  
ReplaceWith | Replace with  
EnterValue | Enter Value  
DefineNameInvalid | The name that you entered is not valid.  
FindValue | Find Value  
ReplaceValue | Replace Value  
DataValidation | Data Validation  
CLEARALL | CLEAR ALL  
APPLY | APPLY  
CellRange | Cell Range  
Allow | Allow  
Data | Data  
Minimum | Minimum  
Maximum | Maximum  
IgnoreBlank | Ignore blank  
WholeNumber | Whole Number  
Decimal | Decimal  
Date | Date  
TextLength | Text Length  
List | List  
NotBetween | Not between  
EqualTo | Equal to  
NotEqualTo | Not equal to  
Greaterthan | Greater than  
Lessthan | Less than

GreaterThanOrEqaulTo | Greater than or eqaul to

LessThanOrEqualTo | Less than or equal to

InCellDropDown | In-cell-dropdown

Sources | Sources

Value | Value

Retry | Retry

DialogError | The list source must be a reference to a single row or column.

ValidationError | This value doesn't match the data validation restrictions defined for the cell.

HideRow | Hide Row

HideRows | Hide Rows

UnHideRows | UnHide Rows

HideColumn | Hide Column

HideColumns | Hide Columns

UnHideColumns | UnHide Columns

InsertRow | Insert Row

InsertRows | Insert Rows

InsertColumn | Insert Column

InsertColumns | Insert Columns

DeleteRow | Delete Row

DeleteRows | Delete Rows

DeleteColumn | Delete Column

DeleteColumns | Delete Columns

Borders | Borders

TopBorders | Top Borders

LeftBorders | Left Borders

RightBorders | Right Borders

BottomBorders | Bottom Borders

AllBorders | All Borders

HorizontalBorders | Horizontal Borders

VerticalBorders | Vertical Borders

OutsideBorders | Outside Borders

InsideBorders | Inside Borders

NoBorders | No Borders

BorderColor | Border Color

BorderStyle | Border Style

INTERCEPT | Calculates the point of the Y-intercept line via linear regression.

SLOPE | Returns the slope of the line from linear regression of the data points.

FreezePanels | Freeze Panels

FreezeRows | Freeze Rows

FreezeColumns | Freeze Columns

UnfreezePanels | Unfreeze Panels

UnfreezeRows | Unfreeze Rows

UnfreezeColumns | Unfreeze Columns

MergeCells | Merge Cells

MergeAll | Merge All

MergeHorizontally | Merge Horizontally

MergeVertically | Merge Vertically

Unmerge | Unmerge

UnmergeCells | Unmerge Cells

SortAscending | Ascending

SortDescending | Descending

CustomSort | Custom Sort

ClearAllFilter | Clear

ReapplyFilter | Reapply

Clear | Clear

ClearContents | Clear Contents

ClearAll | Clear All

ClearFormats | Clear Formats

ClearHyperlinks | Clear Hyperlinks

Image | Image

AddColumn | Add Column

SortBy | Sort by

ThenBy | Then by

Chart | Chart

Column | Column

Bar | Bar



Area | Area

Pie | Pie

Doughnut | Doughnut

PieAndDoughnut | Pie/Doughnut

Line | Line

Radar | Radar

Scatter | Scatter

ChartDesign | Chart Design

ClusteredColumn | Clustered Column

StackedColumn | Stacked Column

StackedColumn100 | 100% Stacked Column

ClusteredBar | Clustered Bar

StackedBar | Stacked Bar

StackedBar100 | 100% Stacked Bar

StackedArea | Stacked Area

StackedArea100 | 100% Stacked Area

StackedLine | Stacked Line

StackedLine100 | 100% Stacked Line

AddChartElement | Add Chart Element

Axes | Axes

AxisTitle | Axis Title

ChartTitle | Chart Title

DataLabels | Data Labels

Gridlines | Gridlines

Legends | Legends

PrimaryHorizontal | Primary Horizontal

PrimaryVertical | Primary Vertical

None | None

AboveChart | Above Chart

Center | Center

InsideEnd | Inside End

InsideBase | Inside Base

OutsideEnd | OutSide End

PrimaryMajorHorizontal | Primary Major Horizontal  
PrimaryMajorVertical | Primary Major Vertical  
PrimaryMinorHorizontal | Primary Minor Horizontal  
PrimaryMinorVertical | Primary Minor Vertical  
Right | Right  
Left | Left  
Bottom | Bottom  
Top | Top  
SwitchRowColumn | Switch Row/Column  
ChartTheme | Chart Theme  
ChartType | Chart Type  
Material | Material  
Fabric | Fabric  
Bootstrap | Bootstrap  
HighContrastLight | HighContrastLight  
MaterialDark | MaterialDark  
FabricDark | FabricDark  
HighContrast | HighContrast  
BootstrapDark | BootstrapDark  
Bootstrap4 | Bootstrap4  
VerticalAxisTitle | Vertical Axis Title  
HorizontalAxisTitle | Horizontal Axis Title  
EnterTitle | Enter Title  
ProtectWorkbook | Protect Workbook  
Password | Password (optional) |  
unProtectPassword | Password  
EnterThePassword | Enter the password  
ConfirmPassword | Confirm Password  
EnterTheConfirmPassword | Re-enter your password  
PasswordAlert | Confirmation password is not identical  
UnProtectWorkbook | Unprotect Workbook  
UnProtectPasswordAlert | The password you supplied is not correct.  
InCorrectPassword | Unable to open the file or worksheet with the given password.

PasswordAlertMsg | Enter the password

ConfirmPasswordAlertMsg | Enter the confirm password

IsProtected | is protected

#### *Loading translations*

To load translation object in an application, use `load` function of the `L10n` class.

The following example demonstrates the Spreadsheet in `French` culture. In the below sample we have translated the ribbon tab names and Home tab content (clipboard, cell style).

#### **CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").Locale("fr-CH").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
ej.base.L10n.load({
 'fr-CH': {
 'spreadsheet': {
 'File': 'Fichier',
 'Home': 'Accueil',
 'Insert': 'Insérer',
 'Formulas': 'Formules',
 'Data': 'Les données',
 'View': 'Vue',
 'Cut': 'Coupe',
 'Copy': 'Copie',
 'Paste': 'Pâte',
 'PasteSpecial': 'Pâte spéciale',
 'All': 'Tous les',
 'Values': 'Valeurs',
 'Formats': 'Les formats',
 'Font': 'fonte',
 'FontSize': 'Taille de police',
 'Bold': 'Audacieux',
 'Italic': 'Italique',
 'Underline': 'Souligner',
 'Strikethrough': 'Barré',
 'TextColor': 'Couleur du texte',
 'FillColor': 'La couleur de remplissage',
 'HorizontalAlignment': 'Alignement horizontal',
 'AlignLeft': 'Alignez à gauche',
 'AlignCenter': 'centre',
 'AlignRight': 'Aligner à droite',
 'VerticalAlignment': 'Alignement vertical',
 'AlignTop': 'Aligner en haut',
 'AlignMiddle': 'Aligner le milieu',
 'AlignBottom': 'Aligner le bas',
 'InsertFunction': 'Insérer une fonction',
 'Delete': 'Effacer',
 'Rename': 'Rebaptiser',
 'Hide': 'Cacher',
```

```

 'Unhide': 'Démasquer',
 'NumberFormat': 'Nombre Format',
 }
}
});
</script>

```

### LOCALECONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynnne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
}

```

```

 ViewBag.DefaultData = data;
 return View();
 }

```

### Internationalization

The Internationalization library is used to globalize number, date, and time values in the spreadsheet component.

The following example demonstrates the Spreadsheet in French [ fr-CH] culture. In the below sample we have globalized the Date(Date column), Time(Time column), and Currency(Amount column) formats.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Locale("fr-CH").Created("created").Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
ej.base.setCulture('fr-CH');
ej.base.setCurrencyCode('EUR');
ej.base.L10n.load({
 'fr-CH': {
 'spreadsheet': {
 'File': 'Fichier',
 'Home': 'Accueil',
 'Insert': 'Insérer',
 'Formulas': 'Formules',
 'Data': 'Les données',
 'View': 'Vue',
 'Cut': 'Coupe',
 'Copy': 'Copie',
 'Paste': 'Pâte',
 'PasteSpecial': 'Pâte spéciale',
 'All': 'Tous les',
 'Values': 'Valeurs',
 'Formats': 'Les formats',
 'Font': 'fonte',
 'FontSize': 'Taille de police',
 'Bold': 'Audacieux',
 'Italic': 'Italique',
 'Underline': 'Souligner',
 'Strikethrough': 'Barré',
 'TextColor': 'Couleur du texte',
 'FillColor': 'La couleur de remplissage',
 'HorizontalAlignment': 'Alignement horizontal',
 'AlignLeft': 'Alignez à gauche',
 'AlignCenter': 'centre',
 'AlignRight': 'Aligner à droite',
 'VerticalAlignment': 'Alignement vertical',
 'AlignTop': 'Aligner en haut',
 'AlignMiddle': 'Aligner le milieu',
 'AlignBottom': 'Aligner le bas',

```

```

 'InsertFunction': 'Insérer une fonction',
 'Delete': 'Effacer',
 'Rename': 'Rebaptiser',
 'Hide': 'Cacher',
 'Unhide': 'Démasquer',
 'NumberFormat': 'Nombre Format',
 }
}
});
function created() {
 var spreadsheet =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 //Applies cell and number formatting to specified range of the
active sheet
 spreadsheet.cellFormat({ fontWeight: 'bold', textAlign: 'center',
verticalAlign: 'middle' }, 'A1:F1');
 spreadsheet.numberFormat('$#,##0.00', 'F2:F11');
}
</script>

```

### INTERNATIONALIZATIONCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gerner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 }
}

```

```

 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### Right to left (RTL)

RTL provides an option to switch the text direction and layout of the Spreadsheet component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Spreadsheet, set the [enableRtl](#) to true.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").Locale("ar-
AE").EnableRtl(true).Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<sctipt>
ej.base.L10n.load({
 'ar-AE': {
 'spreadsheet': {
 "File": "ملف",
 "Home": "هم",
 "Insert": "إدراج",
 "Formulas": "الصيغ",
 "View": "معاينة",
 "Data": "البيانات",
 "Cut": "قطع",
 "Copy": "نسخ",
 "Paste": "معجون",
 "PasteSpecial": "لصق خاص",
 "All": "جميع",
 "Values": "القيم",
 "Formats": "شكل",
 "Font": "الخط",
 "FontSize": "حجم الخط",
 "Bold": "جريء",
 "Italic": "مائل",
 "Underline": "أكد",
 "Strikethrough": "يتوسطه",
 "TextColor": "لون الخط",

```

```

 "FillColor": "لون التعبئة",
 "HorizontalAlignment": "المحاذاة الأفقية",
 "AlignLeft": "محاذاة إلى اليسار",
 "AlignCenter": "مركز",
 "AlignRight": "محاذاة إلى اليمين",
 "VerticalAlignment": "محاذاة عمودية",
 "AlignTop": "محاذاة أعلى",
 "AlignMiddle": "محاذاة الأوسط",
 "AlignBottom": "أسفل محاذاة",
 "InsertFunction": "إدراج وظيفة",
 "Delete": "حذف",
 "Rename": "إعادة تسمية",
 "Hide": "إخفاء",
 "Unhide": "ظهار"
 }
}
});
</script>

```

### RTLCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
 Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
 "07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
 Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
 Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
 Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
 "09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gerner", Model= "GTO", Color=
 "Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
 Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
 Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
 Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
 Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
 "7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
 Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
 "12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
 Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
 Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
 "Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
 "18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
 Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
 Amount= "12317.04" },
 }
}

```



```

 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

See Also

- [Localization](#)

## Use Cases

### Collaborative Editing

The collaborative editing support allows you to work at a spreadsheet collaboratively with other users. Multiple users can access to the the same spreadsheet simultaneously.

#### Dependencies

The following dependent script is required to use the collaborative editing support in spreadsheet.

`js

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/aspnet-signalr/1.1.4/signalr.js"></script>
```

,

#### Client configuration

To broadcast the data for every action in the spreadsheet, you need to transfer the data to the server through `send` method in `actionComplete` event and receive the same data by using the `dataReceived` method. In the `dataReceived` method, you need to update the action to the connected clients through `updateAction` method.

The following code example shows **Collaborative Editing** support in the Spreadsheet control.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ActionComplete("actionComplete").Render()
<script>
 // For signalR Hub connection.
 var connection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreads
heethub', {
 skipNegotiation: true,
 transport: signalR.HttpTransportType.WebSockets
}).build();

```

```
function actionComplete(args) {
 connection.send('BroadcastData', JSON.stringify(args));
}
connection.on('dataReceived', (data) => {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var model = JSON.parse(data);
 spreadsheetObj.updateAction(model);
});
connection.start().then(() => {
 console.log('server connected!!!');
}).catch((err) => console.log(err));
</script>
```

### **COLLABORATIVECONTROLLER.CS**

```
public IActionResult Index()
{
 return View();
}
```

#### *Server configuration*

To make the communication between the server to the connected clients and from clients to the server, you need to configure the signalR Hubs using the following code.

```
`js
// For signalR Hub connection

var connection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost
from ASP.NET Core service

skipNegotiation: true,

transport: signalR.HttpTransportType.WebSockets
}).build();
`
```

#### *Hub configuration*

Initially create a ASP.NET Core project and add the hub file for sending and receiving the data between server and clients.

```
`c#
using Microsoft.AspNetCore.SignalR;
using System.Threading.Tasks;
namespace WebApplication.Hubs
{
 public class SpreadsheetHub : Hub
 {

```

```
public async Task BroadcastData(string data)
{
 await Clients.Others.SendAsync("dataReceived", data);
}
}
}
,
```

To configure the SignalR middleware by registering the following service in the `ConfigureServices` method of the `Startup` class.

```
`c#
services.AddSignalR(e =>
{
 e.MaximumReceiveMessageSize = int.MaxValue; // Option to increase message size for inserting image
 feature. By default, SignalR send messages up to 32 KB.
});
,
```

To set up the SignalR routes by calling `MapHub` in the `Configure` method of the `Startup` class.

```
`c#
app.UseEndpoints(endpoints =>
{
 endpoints.MapRazorPages();
 endpoints.MapHub<SpreadsheetHub>("/hubs/spreadsheethub");
});
,
```

For hosting the service, you may use the above code snippet or download and run the [local service](#).

*Prevent the particular action update for collaborative client*

Using the `action` argument from the `actionComplete` event, you can prevent the particular action update for collaborative client.

The following code example shows how to prevent collaborative client from updating the `format` action.

### CSHTML

```
@Html.EJS().Spreadsheet("spreadsheet").ActionComplete("actionComplete").Render()
<script>
 // For signalR Hub connection
 var connection = new
 signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreadsheethub', { // localhost from ASP.NET Core service
```

```

 skipNegotiation: true,
 transport: signalR.HttpTransportType.WebSockets
 }).build();
 function actionComplete(args) {
 if (args.action !== 'format') { // prevent the format action
 connection.send('BroadcastData', JSON.stringify(args)); // send
 the action data to the server
 }
 }
 connection.on('dataReceived', (data) => {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var model = JSON.parse(data);
 spreadsheetObj.updateAction(model);
 });
 connection.start().then(() => {
 console.log('server connected!!!');
 }).catch((err) => console.log(err));
</script>

```

### COLLABORATIVEPREVENTCONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

#### *Perform import action for collaborative clients*

Using the `action` argument from the `actionComplete` event, you can identify whether the import action is performed or not. If the action is `import`, then you need to send the `response data` to the server and also update the same to the collaborative clients.

The following code example shows how to perform the import functionality for collaborative clients.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ActionComplete("actionComplete").Render()
<script>
 // For signalR Hub connection
 var connection = new
signalR.HubConnectionBuilder().withUrl('https://localhost:44385/hubs/spreads
heethub', { // localhost from ASP.NET Core service
 skipNegotiation: true,
 transport: signalR.HttpTransportType.WebSockets
 }).build();
 function actionComplete(args) {
 if (args.action !== 'format') { // prevent the format action
 connection.send('BroadcastData', JSON.stringify(args)); // send
 the action data to the server
 }
 else {
 // Send the action data to the server for other than import
 actions.
 connection.send("BroadcastData", JSON.stringify(args));

```

```

 }
 }
 connection.on('dataReceived', (data) => {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 var model = JSON.parse(data);
 if (model['action'] === undefined || '') {
 // Load the imported excel file data as JSON to the connected
clients.
 var jsonData = { Workbook: model };
 spreadsheetObj.openFromJson({ file: jsonData });
 }
 else {
 // Update the action details to the connected clients.
 spreadsheetObj.updateAction(model);
 }
 spreadsheetObj.updateAction(model);
 });
 connection.start().then(() => {
 console.log('server connected!!!');
 }).catch((err) => console.log(err));
</script>

```

### COLLABORATIVEIMPORTCONTROLLER.CS

```

public IActionResult Index()
{
 return View();
}

```

## How To

### Sort a range by custom list

You can also define the sorting of cell values based on your own customized personal list. In this article, custom list is achieved using **custom sort comparer**.

In the following demo, the **Trustworthiness** column is sorted based on the custom lists **Perfect**, **Sufficient**, and **Insufficient**.

### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").SortComplete("sortComplete").DataBound("dataBound").AllowSorting(true).Sheets((sheet) =>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.defaultData).Add();
 }).Add();
}).Render()
<script>
 function dataBound() {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (spreadsheetObj.activeSheetIndex === 0) {
 spreadsheetObj.sort({ sortDescriptors: { field: 'C',
sortComparer: mySortComparer }, containsHeader: true }, 'A1:H20');

```

```

 }
}
function sortComplete(args) {
 spreadsheet.selectRange(args.range);
 // code here.
}
// custom sort comparer to sort based on the custom list.
var customList = ['Pink', 'Aquamarine', 'Blue'];
function mySortComparer(x, y) {
 var comparer = ej.data.DataUtil.fnSort('Ascending');
 return comparer(x ? customList.indexOf(x.value) : x, y ?
customList.indexOf(y.value) : y);
};
</script>

```

### CUSTOMSORTCONTROLLER.CS

```

public IActionResult Index()
{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Green", PaymentMode= "Cash On Delivery", DeliveryDate=
"7/01/2017", Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Indigo", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 }
}

```

```

 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

### See Also

- [Filtering](#)
- [Sorting](#)
- [Hyperlink](#)

### Print in Spreadsheet Control

You can use the `print` method by importing from `ej2-base` package. Here, the `Select` event in the dropdown and the `dataBound` event are used to print the single/multiple sheets of data. To print the single/multiple sheets, use the dropdown button and select the `Print` (or) `Print All` option. In the following sample, you can be able to print the single/multiple sheets.

### CSHTML

```

@Html.EJS().DropDownButton("element").Content("Print").Items((IEnumerable<object>)ViewBag.items).Select("itemSelect").Render()
@Html.EJS().Spreadsheet("spreadsheet").DataBound("dataBound").Created("created") Sheets(sheet =>
{
 sheet.Name("Budget").IsProtected(true).Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.budgetData).StartCell("A1").Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
 sheet.Name("Salary").Ranges(ranges =>
 {
 ranges.DataSource((IEnumerable<object>)ViewBag.salaryData).StartCell("A1").Add();
 }).Columns(column =>
 {
 column.Width(100).Add();
 column.Width(100).Add();
 });
}

```

```

 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 var printElement = createElement("div", {
 className: "e-sheet-panel",
 innerHTML:
 '<div class="e-spreadsheet-print"></div><div class="e-sheet"><div
class="e-main-panel style="height:100%" style="overflow: unset"><div
class="e-sheet-content" ></div></div></div>'
 }); // creating same hierarchy of element as DOM
 var isPrint = false;
 function dataBound() {
 if (isPrint) {
 printElement.querySelector(".e-sheet-content").innerHTML += document
 .querySelector(".e-sheet-content").outerHTML;
 var usedRange = this.getActiveSheet().usedRange;
 var tbody = printElement.querySelector('.e-sheet-
content').children[this.activeSheetIndex].querySelector('tbody');
 for (var i = tbody.getElementsByClassName('e-row').length; i >= 0; i--
) {
 if (tbody.getElementsByClassName('e-row')[i] &&
 parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex')) > usedRange.rowIndex + 1) {
 tbody.getElementsByClassName('e-row')[i].remove();
 }
 }
 var sheets = this.sheets;
 if (sheets.length - 1 === this.activeSheetIndex) {
 var count = printElement.querySelector(".e-sheet-
content").childElementCount;
 if (count > 1) {
 for (var i = 0; i < count; i++) {
 (printElement.querySelector('.e-sheet-
content').children[i].getElementsByClassName('e-virtualtrack')[0] as
HTMLElement).style.height = 'auto';
 printElement.querySelector('.e-sheet-
content').children[i].setAttribute('style', 'page-break-after: always;')
 }
 }
 print(printElement);
 isPrint = false;
 printElement.querySelector(".e-sheet-content").innerHTML = '';
 } else {
 if (sheets[this.activeSheetIndex + 1]) {
 this.goTo(sheets[this.activeSheetIndex + 1].name + "!A1");
 }
 }
 }
 }
 function itemSelect(args) {
 if (args.item.text === 'Print') {
 printElement.querySelector(".e-sheet-content").innerHTML =
document.querySelector(
 ".e-sheet-content"
).outerHTML; // To add the spreadsheet table

```



```

 debugger
 var usedRange = this.getActiveSheet().usedRange;
 var tbody = printElement.querySelector('tbody');
 for (var i = tbody.getElementsByClassName('e-row').length; i >=
0; i--) {
 if (tbody.getElementsByClassName('e-row')[i] &&
parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex')) > usedRange.rowIndex + 1) {
 tbody.getElementsByClassName('e-row')[i].remove();
 }
 }
 (printElement.querySelector('.e-sheet-
content').children[0].getElementsByClassName('e-virtualtrack')[0] as
HTMLElement).style.height = 'auto';
 print(printElement);
 printElement.querySelector(".e-sheet-content").innerHTML = '';
 if (args.item.text === 'Print All') {
 var sheets = this.sheets;
 if (this.activeSheetIndex === 0) {
 printElement.querySelector(".e-sheet-content").innerHTML =
document.querySelector(
 ".e-sheet-content"
).outerHTML; // To add the spreadsheet table
 var usedRange = this.getActiveSheet().usedRange;
 var tbody = printElement.querySelector('tbody');
 for (var i = tbody.getElementsByClassName('e-row').length; i
>= 0; i--) {
 if (tbody.getElementsByClassName('e-row')[i] &&
parseInt(tbody.getElementsByClassName('e-row')[i].getAttribute('aria-
rowindex')) > usedRange.rowIndex + 1) {
 tbody.getElementsByClassName('e-row')[i].remove();
 }
 }
 if (sheets[this.activeSheetIndex + 1]) {
 this.goTo(sheets[this.activeSheetIndex + 1].name + "!A1");
 isPrint = true;
 } else {
 print(printElement);
 printElement.querySelector(".e-sheet-content").innerHTML =
'';
 }
 } else {
 if (sheets[0]) {
 this.goTo(sheets[0].name + "!A1");
 isPrint = true;
 }
 }
 }
 function created() {
 this.cellFormat({ fontWeight: 'bold', fontSize: '12pt'}, 'A1:E1');
 this.cellFormat({ color: '#10c469' }, 'B1:B10');
 }
 </script>

```

**PRINTCONTROLLER.CS**

```

public IActionResult Index()
{
 List<object> data1 = new List<object>()
 {
 new { ExpenseType= "Housing", ProjectedCost= "7000",
ActualCost= "7500", Difference= "-500"},
 new { ExpenseType= "Transportation", ProjectedCost= "500",
ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Insurance", ProjectedCost= "1000",
ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Food", ProjectedCost= "2000",
ActualCost= "1800", Difference= "200"},
 new { ExpenseType= "Pets", ProjectedCost= "300",
ActualCost= "200", Difference= "100"},
 new { ExpenseType= "Personel Care", ProjectedCost= "500",
ActualCost= "500", Difference= "0"},
 new { ExpenseType= "Loan", ProjectedCost= "1000",
ActualCost= "1000", Difference= "0"},
 new { ExpenseType= "Tax", ProjectedCost= "200", ActualCost=
"200", Difference= "0"},
 new { ExpenseType= "Savings", ProjectedCost= "1000",
ActualCost= "900", Difference= "100"},
 new { ExpenseType= "Total", ProjectedCost= "13500",
ActualCost= "13600", Difference= "-100"},
 };
 List<object> data2 = new List<object>()
 {
 new { Earnings= "Basic", CreditAmount= "20000",
Deductions= "Provident Fund", DebitAmount= "2400"},
 new { Earnings= "HRA", CreditAmount= "8000", Deductions=
"ESI", DebitAmount= "0"},
 new { Earnings= "Special Allowance", CreditAmount= "25000",
Deductions= "Professional Tax", DebitAmount= "200"},
 new { Earnings= "Incentives", CreditAmount= "2000",
Deductions= "TDS", DebitAmount= "2750"},
 new { Earnings= "Bonus", CreditAmount= "1500", Deductions=
"Other Deduction", DebitAmount= "0"},
 new { Earnings= "Total Earnings", CreditAmount= "56500",
Deductions= "Total Deductions", DebitAmount= "5350"},
 };
 List<object> items = new List<object>();
 items.Add(new
 {
 text = "Print"
 });
 items.Add(new
 {
 text = "PrintAll"
 });
 ViewBag.items = items;
 ViewBag.budgetData = data1;
 ViewBag.salaryData = data2;
 return View();
}

```

*Changing the active sheet while importing a file in Spreadsheet Control*

You can change the active sheet of imported file by updating [activeSheetIndex](#) property on the [openComplete](#) event.

The following code example shows how to set the active sheet when importing an Excel file.

**CSHTML**

```
@Html.EJS().Spreadsheet("spreadsheet").OpenUrl("Open").OpenComplete("openComplete").Render()
<script>
 function openComplete(args) {
 var spreadsheetObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (spreadsheetObj) {
 spreadsheetObj.activeSheetIndex = 2;
 }
 }
</script>
```

**OPENCONTROLLER.CS**

```
public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}
```

*Import an excel document using file uploader in ASP.NET MVC Spreadsheet control*

If you explore your machine to select and upload an excel document using the file uploader, you will receive the uploaded document as a raw file in the **success** event of the file uploader. In this **success** event, you should pass the received raw file as an argument to the Spreadsheet's **open** method to see the appropriate output.

The following code example shows how to import an excel document using file uploader in spreadsheet.

**CSHTML**

```
@Html.EJS().Uploader("UploadFiles").Success("onSuccess").AllowedExtensions(".xls, .xlsx, .csv").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
@Url.Content("Home/Save"), RemoveUrl = @Url.Content("Home/Remove")
}).Render()
@Html.EJS().Spreadsheet("spreadsheet").OpenUrl("Home/Open").Render()
<script>

 function onSuccess(args) {
 var ssObj =
ej.base.getComponent(document.getElementById('spreadsheet'), 'spreadsheet');
 if (args.operation === 'upload')
 ssObj.open({ file: args.file.rawFile });
 }
</script>
```

**OPENCONTROLLER.CS**

```

public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}

public void Save(IList<IFormFile> UploadFiles)
{
 long size = 0;
 try
 {
 foreach (var file in UploadFiles)
 {
 var filename = ContentDispositionHeaderValue
 .Parse(file.ContentDisposition)
 .FileName
 .Trim('"');
 filename = hostingEnv.WebRootPath + $"\"{filename}\"";
 size += file.Length;
 if (!System.IO.File.Exists(filename))
 {
 using (FileStream fs =
System.IO.File.Create(filename))
 {
 //file.CopyTo(fs);
 //fs.Flush();
 }
 }
 else
 {
 using (FileStream fs = System.IO.File.Open(filename,
FileMode.Append))
 {
 //file.CopyTo(fs);
 //fs.Flush();
 }
 }
 }
 }
 catch (Exception e)
 {
 Response.Clear();
 Response.StatusCode = 204;

 Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
 "File failed to upload";

 Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
 e.Message;
 }
}

public void Remove(string UploadFile)
{
 try
 {

```

```

var filename = hostingEnv.WebRootPath + $"{@"\{UploadFile}"}";
if (System.IO.File.Exists(filename))
{
 System.IO.File.Delete(filename);
}
catch (Exception e)
{
 Response.Clear();
 Response.StatusCode = 200;

 Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
 "File removed successfully";

 Response.HttpContext.Features.Get<IHttpResponseFeature>().ReasonPhrase =
 e.Message;
}
}

```

#### Identify the context menu opened in Spreadsheet control

The Spreadsheet includes several context menus that will open and display depending on the action. When you right-click on a cell, for example, a context menu with options related to the cell element appears.

The class name returned by the [contextMenuBeforeOpen](#) event can be used to identify the context menu that is opened. The context menus and their class names are tabulated below.

| Class name       | Context menu name          |
|------------------|----------------------------|
| ----- -----      |                            |
| .e-sheet-content | Cell context menu          |
| .e-toolbar-item  | Footer context menu        |
| .e-rowhdr-table  | Row header context menu    |
| .e-colhdr-table  | Column header context menu |

The following code example shows how to identify the context menu opened.

#### CSHTML

```

@Html.EJS().Spreadsheet("spreadsheet").ContextMenuBeforeOpen("ContextMenuBeforeOpen").Render()
<script>
 function ContextMenuBeforeOpen(args) {
 if (ejs.base.closest(args.event.target, '.e-sheet-content')) {
 console.log('Cell Context Menu');
 } else if (ejs.base.closest(args.event.target, '.e-colhdr-table')) {
 console.log('Column Header Context Menu');
 } else if (ejs.base.closest(args.event.target, '.e-rowhdr-table')) {
 console.log('Row Header Context Menu');
 } else if (ejs.base.closest(args.event.target, '.e-toolbar-item')) {
 console.log('Footer Context Menu');
 }
 }
</script>

```

### Save and open Spreadsheet data as a Base64 string in ASP.NET MVC Spreadsheet control

In the Spreadsheet control, there is currently no direct option to save and open data as a **Base64** string. You can achieve this by saving the Spreadsheet data as blob data and then converting that saved blob data to a **Base64** string using **FileReader**.

You can get the Spreadsheet data as blob in the [saveComplete](#) event when you set the **needBlobData** as **true** and **isFullPost** as **false** in the [beforeSave](#) event.

The following code example shows how to save and open the spreadsheet data as base64 string.

#### CSHTML

```
@Html.EJS().Button("importBtn").Content("Import Base64").Render();
@Html.EJS().Button("exportBtn").Content("Export as Base64").Render();
@Html.EJS().Spreadsheet("spreadsheet").BeforeSave("beforeSave").SaveComplete(
 "saveComplete").Sheets(sheet => {
 sheet.Name("Price Details").Ranges(ranges => {

ranges.DataSource((IEnumerable<object>)ViewBag.DefaultData).StartCell("A1").
Add();
 }).Columns(column => {
 column.Width(130).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 var base64String;
 function beforeSave(args) {
 args.needBlobData = true; // To trigger the saveComplete event.
 args.isFullPost = false; // Get the spreadsheet data as blob data in
the saveComplete event.
 }
 function saveComplete(args) {
 // Convert blob data to base64 string.
 var reader = new FileReader();
 reader.readAsDataURL(args.blobData);
 reader.onloadend = function () {
 base64String = reader.result;
 };
 }
 document.getElementById("importBtn").addEventListener('click', function
() {
 var spreadsheetObj =
document.getElementById("spreadsheet").ej2_instances[0];
 // Open the file based on saved base64 string.
 fetch(base64String)
 .then((response) => response.blob())
 .then((fileBlob) => {
 var file = new File([fileBlob], 'Sample.xlsx');
 spreadsheetObj.open({ file: file });
 });
 });
 document.getElementById("exportBtn").addEventListener('click', function
() {
```

```

var spreadsheetObj =
document.getElementById("spreadsheet").ej2_instances[0];
spreadsheetObj.save({
 url: 'Home/Save',
 fileName: 'Worksheet',
 saveType: 'Xlsx',
}); // Specifies the save URL, file name, file type need to be
saved.
// Logs base64 string into the console.
console.log('Base64 String - ', base64String);
});
</script>

```

### OPENCONTROLLER.CS

```

public IActionResult Open(IFormCollection openRequest)
{
 OpenRequest open = new OpenRequest();
 open.File = openRequest.Files[0];
 return Content(Workbook.Open(open));
}

public void Save(SaveSettings saveSettings)
{
 Workbook.Save(saveSettings);
}

public IActionResult Index()
{
 List<object> defaultData = new List<object>()
 {
 new { Item Name= "Casual Shoes", Date= "02/14/2014", Time=
"11=34=32 AM", Quantity= "10", Price= "20", Amount= "200", Discount= "1",
Profit= "10" },
 new { Item Name= "Sports Shoes", Date= "06/11/2014", Time=
"05=56=32 AM", Quantity= "20", Price= "30", Amount= "600", Discount= "5",
Profit= "50" },
 new { Item Name= "Formal Shoes", Date= "07/27/2014", Time=
"03=32=44 AM", Quantity= "20", Price= "15", Amount= "300", Discount= "7",
Profit= "27" },
 new { Item Name= "Sandals & Floaters", Date= "11/21/2014",
Time= "06=23=54 AM", Quantity= "15", Price= "20", Amount= "300", Discount=
"11", Profit= "67" },
 new { Item Name= "Flip- Flops & Slippers", Date=
"06/23/2014", Time= "12=43=59 AM", Quantity= "30", Price= "10", Amount=
"300", Discount= "10", Profit= "70" },
 new { Item Name= "Sneakers", Date= "07/22/2014", Time=
"10=55=53 AM", Quantity= "40", Price= "20", Amount= "800", Discount= "13",
Profit= "66" },
 new { Item Name= "Running Shoes", Date= "02/04/2014", Time=
"03=44=34 AM", Quantity= "20", Price= "10", Amount= "200", Discount= "3",
Profit= "14" },
 new { Item Name= "Loafers", Date= "11/30/2014", Time=
"03=12=52 AM", Quantity= "31", Price= "10", Amount= "310", Discount= "6",
Profit= "29" },
 new { Item Name= "Cricket Shoes", Date= "07/09/2014", Time=
"11=32=14 AM", Quantity= "41", Price= "30", Amount= "1210", Discount= "12",
Profit= "166" },
 }
}

```

```

 new { Item Name= "T-Shirts", Date= "10/31/2014", Time=
"12=01=44 AM", Quantity= "50", Price= "10", Amount= "500", Discount= "9",
Profit= "55" }
 };
 ViewBag.DefaultData = defaultData;
 return View();
}

```

### *Insert a sheet programmatically and make it the active sheet in ASP.NET MVC Spreadsheet control*

A sheet is a collection of cells organized in the form of rows and columns that allows you to store, format, and manipulate the data. Using `insertSheet` method, you can insert one or more sheets at the desired index. Then, you can make the inserted sheet as active sheet by focusing the start cell of that sheet using the `goTo` method.

The following code example shows how to insert a sheet programmatically and make it the active sheet.

### **CSHTML**

```

@Html.EJS().Button("insertSheet").Content("Insert Sheet").Render();
@Html.EJS().Spreadsheet("spreadsheet").Sheets(sheet => {
 sheet.Name("Price Details").Ranges(ranges => {
 ranges.DataSource((IEnumerable < object
>)ViewBag.DefaultData).StartCell("A1").Add();
 }).Columns(column => {
 column.Width(130).Add();
 column.Width(100).Add();
 column.Width(100).Add();
 }).Add();
}).Render()
<script>
 document.getElementById("insertSheet").addEventListener('click',
function () {
 var spreadsheetObj =
document.getElementById("spreadsheet").ej2_instances[0];
 spreadsheetObj.insertSheet(
 [
 {
 index: 1,
 name: 'new_sheet',
 ranges: [
 {
 dataSource: ViewBag.ProductData,
 startCell: 'A1'
 },
],
 },
],
);
 // Use the timeout function to wait until the sheet is inserted.
 setTimeout(() => {
 // Method for switching to a new sheet.
 spreadsheet.goTo('new_sheet!A1');
 })
});
</script>

```



**INSERTSHEETCONTROLLER.CS**

```

public IActionResult Index()
{
 List<object> defaultData = new List<object>()
 {
 new { Item Name= "Casual Shoes", Date= "02/14/2014", Time=
"11=34=32 AM", Quantity= "10", Price= "20", Amount= "200", Discount= "1",
Profit= "10" },
 new { Item Name= "Sports Shoes", Date= "06/11/2014", Time=
"05=56=32 AM", Quantity= "20", Price= "30", Amount= "600", Discount= "5",
Profit= "50" },
 new { Item Name= "Formal Shoes", Date= "07/27/2014", Time=
"03=32=44 AM", Quantity= "20", Price= "15", Amount= "300", Discount= "7",
Profit= "27" },
 new { Item Name= "Sandals & Floaters", Date= "11/21/2014",
Time= "06=23=54 AM", Quantity= "15", Price= "20", Amount= "300", Discount=
"11", Profit= "67" },
 new { Item Name= "Flip- Flops & Slippers", Date=
"06/23/2014", Time= "12=43=59 AM", Quantity= "30", Price= "10", Amount=
"300", Discount= "10", Profit= "70" },
 new { Item Name= "Sneakers", Date= "07/22/2014", Time=
"10=55=53 AM", Quantity= "40", Price= "20", Amount= "800", Discount= "13",
Profit= "66" },
 new { Item Name= "Running Shoes", Date= "02/04/2014", Time=
"03=44=34 AM", Quantity= "20", Price= "10", Amount= "200", Discount= "3",
Profit= "14" },
 new { Item Name= "Loafers", Date= "11/30/2014", Time=
"03=12=52 AM", Quantity= "31", Price= "10", Amount= "310", Discount= "6",
Profit= "29" },
 new { Item Name= "Cricket Shoes", Date= "07/09/2014", Time=
"11=32=14 AM", Quantity= "41", Price= "30", Amount= "1210", Discount= "12",
Profit= "166" },
 new { Item Name= "T-Shirts", Date= "10/31/2014", Time=
"12=01=44 AM", Quantity= "50", Price= "10", Amount= "500", Discount= "9",
Profit= "55" }
 };
 ViewBag.DefaultData = defaultData;
 List<object> data = new List<object>()
 {
 new { ProductId= "1", ProductName= "Chai", SupplierId=
"1", QuantityPerUnit= "10 boxes x 20 bags", UnitPrice= "18.00",
UnitsInStock= "39", Discontinued= "false" },
 new { ProductId= "2", ProductName= "Chang", SupplierId=
"1", QuantityPerUnit= "24 - 12 oz bottles", UnitPrice= "19.00",
UnitsInStock= "17", Discontinued= "true" },
 new { ProductId= "3", ProductName= "Aniseed Syrup",
SupplierId= "1", QuantityPerUnit= "12 - 550 ml bottles", UnitPrice=
"10.00", UnitsInStock= "13", Discontinued= "false" },
 new { ProductId= "4", ProductName= "Chef Anton\'s Cajun
Seasoning", SupplierId= "2", QuantityPerUnit= "48 - 6 oz jars",
UnitPrice= "22.00", UnitsInStock= "53", Discontinued= "true" },
 new { ProductId= "5", ProductName= "Chef Anton\'s Gumbo
Mix", SupplierId= "2", QuantityPerUnit= "36 boxes", 'Unit Price',
UnitPrice= "21.35", UnitsInStock= "0", Discontinued= "true" },
 }
}

```

```

 new { ProductId= "6", ProductName= "Grandma\'s Boysenberry
Spread", SupplierId= "3", QuantityPerUnit= "12 - 8 oz jars", UnitPrice=
"25.00", UnitsInStock= "120", Discontinued= "false" },
 new { ProductId= "7", ProductName= "Uncle Bob\'s Organic
Dried Pears", SupplierId= "3", QuantityPerUnit= "12 - 1 lb pkgs.",
UnitPrice= "30.00", UnitsInStock= "15", Discontinued= "true" },
 new { ProductId= "10", ProductName= "Queso Cabrales",
SupplierId= "5", QuantityPerUnit= "1 kg pkg.", UnitPrice= "21.00",
UnitsInStock= "22", Discontinued= "false" },
 };
 ViewBag.ProductData = data;
 return View();
}

```

#### Get the filtered rows in ASP.NET MVC Spreadsheet control

Filtering allows you to view specific rows in a spreadsheet while hiding the others. The [allowFiltering](#) property allows you to enable or disable filtering functionality through the UI. You can also use the [allowFiltering](#) property and [applyFilter](#) method combination to filter data via code behind. The filtered rows can be identified by iterating through the row collection on the sheet and using the [isFiltered](#) property available in each row object.

The following code example shows how to get the filtered rows.

#### CSHTML

```

@Html.EJS().Button("getFilterData").Content("Get Filtered Data").Render();
@Html.EJS().Spreadsheet("spreadsheet").AllowFiltering(true).Sheets((sheet)
=>
{
 sheet.Ranges((ranges) =>
 {
 ranges.DataSource(@ViewBag.DefaultData).Add();
 }).Add();
}).Render()
<script>
 document.getElementById("getFilterData").addEventListener('click',
function () {
 var spreadsheet =
document.getElementById("spreadsheet").ej2_instances[0];
 var activeSheet = spreadsheet.getActiveSheet();
 var usedRange = activeSheet.usedRange;
 for (var i = 0; i <= usedRange.rowIndex; i++) {
 // Get the filtered row using isFiltered property.
 var filteredRow = (activeSheet.rows[i]).isFiltered;
 if (!filteredRow) {
 var rowData = spreadsheet.getRowData(i);
 console.log("Row:", i + 1, "Cells", rowData);
 }
 }
 });
</script>

```

#### INSERTSHEETCONTROLLER.CS

```

public IActionResult Index()

```

```

{
 List<object> data = new List<object>()
 {
 new { CustomerName= "Romona Heaslip", Model= "Taurus",
Color= "Aquamarine", PaymentMode= "Debit Card", DeliveryDate=
"07/11/2015", Amount= "8529.22" },
 new { CustomerName= "Clare Batterton", Model= "Sparrow",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/13/2016",
Amount= "17866.19" },
 new { CustomerName= "Eamon Traise", Model= "Grand
Cherokee", Color= "Blue", PaymentMode= "Net Banking", DeliveryDate=
"09/04/2015", Amount= "13853.09" },
 new { CustomerName= "Julius Gorner", Model= "GTO", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/15/2017",
Amount= "2338.74" },
 new { CustomerName= "Jenna Schoolfield", Model= "LX",
Color= "Yellow", PaymentMode= "Credit Card", DeliveryDate= "10/08/2014",
Amount= "9578.45" },
 new { CustomerName= "Marylynne Harring", Model= "Catera",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate= "7/01/2017",
Amount= "19141.62" },
 new { CustomerName= "Vilhelmina Leipelt", Model= "7
Series", Color= "Goldenrod", PaymentMode= "Credit Card", DeliveryDate=
"12/20/2015", Amount= "6543.30" },
 new { CustomerName= "Barby Heisler", Model= "Corvette",
Color= "Red", PaymentMode= "Credit Card", DeliveryDate= "11/24/2014",
Amount= "13035.06" },
 new { CustomerName= "Karyn Boik", Model= "Regal", Color=
"Pink", PaymentMode= "Debit Card", DeliveryDate= "05/12/2014", Amount=
"18488.80" },
 new { CustomerName= "Jeanette Pamplin", Model= "S4",
Color= "Fuscia", PaymentMode= "Net Banking", DeliveryDate= "12/30/2014",
Amount= "12317.04" },
 new { CustomerName= "Cristi Espinos", Model= "TL", Color=
"Aquamarine", PaymentMode= "Credit Card", DeliveryDate= "12/18/2013",
Amount= "6230.13" },
 new { CustomerName= "Issy Humm", Model= "Club Wagon",
Color= "Pink", PaymentMode= "Cash On Delivery", DeliveryDate=
"02/02/2015", Amount= "9709.49" },
 new { CustomerName= "Tuesday Fautly", Model= "V8 Vantage",
Color= "Crimson", PaymentMode= "Debit Card", DeliveryDate= "11/19/2014",
Amount= "9766.10" },
 new { CustomerName= "Rosemaria Thomann", Model= "Caravan",
Color= "Violet", PaymentMode= "Net Banking", DeliveryDate= "02/08/2014",
Amount= "7685.49" },
 };
 ViewBag.DefaultData = data;
 return View();
}

```

## Mobile Responsiveness

The Spreadsheet control rendered in desktop mode will be adaptive in all mobile devices where the layout gets adjusted based on their parent element's dimensions to accommodate any resolution.

You can see the overflowed items of ribbon header, ribbon content, and sheet tab using touch and swipe action. The right navigation arrow is added at the end of the ribbon content through which the user can navigate towards overflowed items. Once you reached the rightmost end of the ribbon content, the right navigation arrow will change to left navigation arrow through which you can navigate to the left of the ribbon content.



Comparision between EJ1 & EJ2 Spreadsheet features

The following table compares Excel functionality with the availability of EJ1 and EJ2 Spreadsheet features.

| Features    | Available in EJ1 Spreadsheet | Available in EJ2 Spreadsheet | Comments |
|-------------|------------------------------|------------------------------|----------|
| ---         | ---                          | ---                          | ---      |
| Ribbon      | Yes                          | Yes                          | -        |
| Formula bar | Yes                          | Yes                          | -        |
| Sheet tab   | Yes                          | Yes                          | -        |

|                                  |           |           |                                                                   |  |
|----------------------------------|-----------|-----------|-------------------------------------------------------------------|--|
| Show / Hide gridlines and header | Yes       | Yes       | -                                                                 |  |
| Scrolling                        | Partially | Yes       | -                                                                 |  |
| Selection                        | Yes       | Yes       | -                                                                 |  |
| Editing                          | Yes       | Yes       | -                                                                 |  |
| Formulae                         | Yes       | Partially | EJ2 supports limited number of <a href="#">most used formulas</a> |  |
| Named range                      | Yes       | Partially | EJ2 Spreadsheet Named range supports only in workbook scope       |  |
| Data Binding                     | Yes       | Yes       | -                                                                 |  |
| Formatting                       | Yes       | Yes       | -                                                                 |  |
| Context menu                     | Yes       | Yes       | -                                                                 |  |
| Keyboard navigation              | Yes       | Yes       | -                                                                 |  |
| Keyboard shortcuts               | Yes       | Yes       | -                                                                 |  |
| Sorting                          | Yes       | Yes       | -                                                                 |  |
| Filtering                        | Yes       | Yes       | -                                                                 |  |
| Hyperlink                        | Yes       | Yes       | -                                                                 |  |
| Undo & redo                      | Yes       | Yes       | -                                                                 |  |
| Open and Save                    | Yes       | Yes       | -                                                                 |  |
| Resize / Autofit                 | Yes       | Yes       | -                                                                 |  |
| Clipboard                        | Yes       | Yes       | -                                                                 |  |
| Collaborative editing            | No        | Yes       | -                                                                 |  |
| Wrap text                        | Yes       | Yes       | -                                                                 |  |
| Template                         | No        | Yes       | -                                                                 |  |
| Merge cells                      | Yes       | Yes       | -                                                                 |  |
| Show / Hide rows and columns     | Yes       | Yes       | -                                                                 |  |
| Sheet customizations             | Yes       | Partially | Move or copy sheet is not supported in EJ2 spreadsheet.           |  |
| Data Validation                  | Yes       | Yes       | -                                                                 |  |
| Table                            | Yes       | No        | -                                                                 |  |
| Chart                            | Yes       | Yes       | -                                                                 |  |
| Image                            | Yes       | Yes       | -                                                                 |  |
| Conditional formatting           | Yes       | Yes       | -                                                                 |  |
| Freeze Pane                      | Yes       | Yes       | -                                                                 |  |
| Scaling                          | No        | No        | -                                                                 |  |
| Print                            | Yes       | No        | -                                                                 |  |
| Grouping                         | No        | No        | -                                                                 |  |

| Autofill | Yes | No | - |  
| Auto Sum | Yes | Yes | - |  
| Format painter | Yes | No | - |  
| Cell Style | Yes | Partially | We can only customize the cell style in EJ2 Spreadsheet through API. |  
| Protection | Yes | Partially | Custom encryption is not supported in EJ2 Spreadsheet's protect workbook. |  
| Find and replace | Yes | Yes | - |  
| Drag and Drop | Yes | No | - |  
| Notes | Yes | No | - |  
| Comments | No | No | - |  
| Pivot table | Yes | No | - |  
| Sparklines | Yes | No | - |  
| Form controls | Yes | No | - |  
| Shapes | No | No | - |  
| Clear | Yes | Yes | - |

See Also

- [Cell range](#)
- [Formatting](#)
- [Hyperlink](#)
- [Undo and Redo](#)
- [Unlock the particular cells in the protected sheet](#)

## Stepper

### Getting Started with ASP.NET MVC Stepper Control

This section briefly explains about how to include **ASP.NET MVC Stepper** control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

## PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/bootstrap5.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
```

```
</body>
```

### Add ASP.NET MVC Stepper control

Now, add the Syncfusion ASP.NET MVC Stepper control in `~/Views/Home/Index.cshtml` page.

#### INDEX.CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Render()
```

### Adding steps

You can define steps by setting the `Steps` property.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations
<div id="default-Stepper">
 @Html.EJS().Stepper("stepper").Steps(ViewBag.DefaultStepper).Render()
</div>
```

#### DEFAULT.CS

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 ViewBag.DefaultStepper = defaultStepper;

 return View();
}
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Stepper control will be rendered in the default web browser.



### Configure icon and label

You can define the step icon and label by setting the `IconCss` and `Label` properties using `Steps` property.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconWithLabel).Render()
<style>
 @@font-face {
```



```
font-family: 'Default';
src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGvHhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOCc5wbnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxBgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbYICAgJCQoKCgkKCAk
IBwcHBQEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkKJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICak
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCGsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkKJCAcHBgYFBAM
DAQEBAQMDBAUGBgHCAkKJCQkKCGoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkKJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkKJCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkKJCgGuAQI
GehYJBKYp/10ICAcGBQCAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBGcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OWEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAXEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQgHCPcBAGQGBBgKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHfYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHfYUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQDDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAWMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEESFBUVfHcXfYVFB1SEA8NCwoIBQQAQCFCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgHbwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCgghBQE
BAikCAgIEAwQFDA0SBwcGAgIBAQICBgHbXyKQkKJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHfYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAuUGBwgJCgICAQENAEQFAwI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQIAGEBaf6RDAsLCQkICAYGBQUDAwIBAQIDAuUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQqNEAkKCwNDXAREXQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDAMCAQICAwCYAwEaBwQBAgIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMCFBIIcCsQKaEqRUclJSYnJykpKiosLC4GfgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAABAAcAALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhcR
lZmF1bHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABEAGUAZgBhAHUABAB0AFIAZQB
```

```
nAHUAbABhAHlARABlAGYAYQB1AGWAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHlAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAA
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-user:before {
 content: "\e708";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
}
</style>
```

### ICONWITHLABEL.CS

```
public ActionResult Demo()
{
 List<Step> iconWithLabel = new List<Step>();
 iconWithLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart"
});
 iconWithLabel.Add(new Step { Label = "Address", IconCss = "sf-icon-user"
});
 iconWithLabel.Add(new Step { Label = "Delivery", IconCss = "sf-icon-
transport" });
 iconWithLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-
payment" });
 iconWithLabel.Add(new Step { Label = "Ordered", IconCss = "sf-icon-
success" });
 ViewBag.IconWithLabel = iconWithLabel;

 return View();
}
```



## Steps in ASP.NET MVC Stepper control

The ASP.NET MVC Stepper allows you to add steps using the `Steps` property. Each step can be configured with options such as `IconCss`, `Text`, `Label`, `CssClass` and more.

### Adding steps

You can define the icon and text content for each step using the `IconCss`, `Text` and `Label` properties.

### Defining icon CSS

You can define the CSS class to show the icon for each step using the `IconCss` property.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconOnly).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAANmhoZWEIUAQAAAArAAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1lUf5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AoaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAQAAAAEAAAAAAAAAAgAAAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkXgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAgIBGcFBQQEAgH+CwEBAGQEBQUHBGgICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQCHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBGcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAGQGBgGKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKcAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQDAGECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAqIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKcAYGawMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQAQQAQoLDQ8QEHlUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
```

```
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBaGMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAGICQgHCACHBgYFBQQDAwIBAQIDAQWQBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAIkCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYKcQkJCACHBgUFBAQCAQEBAQIDAQWQBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXFxcWFxYXfHwYFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRI
HBqADChI1DQoFAGEBaGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAUGBwgJCgICAQENAQEFaWI
DAGECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQWUBgYICakJCwsMKSckIiAeGxoYfHq
TERAPDQwLCgkIDxsJBQUBQqNEakKCwWnDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwGAGIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRYbyBTdHVKaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGWAdABEAGUAZgBhAHUAbAB0AFYAZQBByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAIAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAA
AAAAAAYBAGEDAQQBBQEQAQcADXRyYW5zcG9ydC12YW4LdXNlcil1b2RpbmctY2FyY2F8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueype');
```

```
font-weight: normal;
```

```
font-style: normal;
```

```
}
```

```
[class^="sf-icon-"], [class*=" sf-icon-"] {
```

```
font-family: 'Default' !important;
```

```
speak: none;
```

```
font-style: normal;
```

```
font-weight: normal;
```

```
font-variant: normal;
```

```
text-transform: none;
```

```
line-height: inherit;
```

```
-webkit-font-smoothing: antialiased;
```

```
-moz-osx-font-smoothing: grayscale;
```

```
}
```

```
.sf-icon-cart:before {
```

```
content: "\e710";
```

```
}
```

```
.sf-icon-transport:before {
```

```
content: "\e702";
```

```
}
```

```
.sf-icon-payment:before {
```

```
content: "\e706";
```

```
}
```

```
.sf-icon-success:before {
```

```
content: "\e715";
```

```
}
```

```
</style>
```

### ICON.CS

```
public ActionResult Demo()
{
 List<Step> iconOnly = new List<Step>();
 iconOnly.Add(new Step { IconCss = "sf-icon-cart" });
 iconOnly.Add(new Step { IconCss = "sf-icon-transport" });
 iconOnly.Add(new Step { IconCss = "sf-icon-payment" });
 iconOnly.Add(new Step { IconCss = "sf-icon-success" });
 ViewBag.IconOnly = iconOnly;

 return View();
}
```

#### Defining text content

You can define text instead of an icon by setting the [Text](#) property and display label content for a step using the [Label](#) property.

When both label and text are defined, the label takes priority for display based on the [StepType](#).

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("textOnly").Steps(ViewBag.TextOnly).Render()
@Html.EJS().Stepper("labelOnly").Steps(ViewBag.LabelOnly).Render()
```

### TEXT.CS

```
public ActionResult Demo()
{
 List<Step> textOnly = new List<Step>();
 textOnly.Add(new Step { Text = "A" });
 textOnly.Add(new Step { Text = "B" });
 textOnly.Add(new Step { Text = "C" });
 textOnly.Add(new Step { Text = "D" });
 List<Step> labelOnly = new List<Step>();
 labelOnly.Add(new Step { Label = "Cart" });
 labelOnly.Add(new Step { Label = "Delivery Address" });
 labelOnly.Add(new Step { Label = "Payment" });
 labelOnly.Add(new Step { Label = "Confirmation" });

 ViewBag.LabelOnly = labelOnly;
 ViewBag.TextOnly = textOnly;

 return View();
}
```

#### Optional steps

You can show whether the step is optional or not by using [Optional](#) property. By default, the [Optional](#) property is [false](#).

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconOnly).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1lUf5THQAACTgAAAIlcG9zdJ8LuOM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAAA
EAAAAABAAAAQAQAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOCc5wbncOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAAwADAAAAEABAACAAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkXgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWefDjM/BzUnIw8GATM/Dx8PMxEhAg8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAChBgYFBAM
DAQEBAQMDBAUGBgCHCAkKCgkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkKCgoJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkKCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKCgGuAQI
GehYJBKYp/l0ICAcGBQQAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHLWejDwe1LwMjNz0BJzcfBTcfAg8BLwY
3OwEFAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAXEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAQgGBgkKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHfYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXfHfYUFEQDw0LCgGfBAEXBhCFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQDAGECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAgIBGQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcxYFVBISEA8NCwoIBQQBAQQFCAoLDQ8QEHfYFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBQADAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQADAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAGwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAGwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAEBAQICBAQFBQY
GBwCHCAgICQgHCACHBgYFBQQDAwIBAQIDAQWFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWefBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCgghBQE
BAIkCAGIEAwQFDA0SBwcGAgIBAQICBgCHBxYKCQkKJCAChBgUFBAQCAQEBQIDAQWFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIBwYCAQEBQEBAGYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAGwMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHfYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAgQCAQEBAwkRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAIDAwUGBwgJCgICAQENAQEFawI
DAGECAGMFAwMEBAUDAMFAwIBAQECAGwMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAYDIgICAQECAlI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAGwQEAQEBQICBAMFBQUGBwCICAgJBwgHBgc
GBgUFBAQEBQqIAGEBaf6RDAsLCQkICAYGBQUDAwIBAQIDAQWFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQWFBQqNEAkKCwwNDxARExQWGBobHiAiJCCoAMDawQECA8XPRcKCGUPFz0
REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwGAGIwXmMXFBIIcCsKqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNGsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAAAADAACADwABAAAAAEEAACAFgA
BAAAAAAFAAASAHQABAAAAAAGAACAkAABAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI

```

```

AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAA
AAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbnW9uZXkFY2hlY2sAAAA=) format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
.stepper-section {
 margin: 50px 100px;
}
</style>

```

### OPTIONAL.CS

```

public ActionResult Demo()
{
 List<Step> iconOnly = new List<Step>();
 iconOnly.Add(new Step { IconCss = "sf-icon-cart" });
 iconOnly.Add(new Step { IconCss = "sf-icon-transport" });
 iconOnly.Add(new Step { IconCss = "sf-icon-payment", Optional = true });
 iconOnly.Add(new Step { IconCss = "sf-icon-success" });

 ViewBag.IconOnly = iconOnly;

 return View();
}

```



### Disabling steps

You can use the [Disabled](#) property to disable a step, preventing user interaction when set to `true`. By default, the value is `false`.

### CSS/HTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconOnly).Render()

<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAKAIAAAAwAgT1MvMj1vSgcAAAEoAAAAVmNtYXcDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAActgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAQAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAWADAAMAADAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWefDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQAeAgH+CwEBAGQEBQUHBggICak
JCgoKCQoICQgHBwCfBQQAeBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAyGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLcwsJCQkHBwUFAwKE/eMBAQoJCQkJCAChBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLcgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCh/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHlwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQCHCPcBAGQGBgKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHwYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHwYUExEQDw0LCgGFBAEXBhCFBAMDrxYWDQEBawUHCAmsMDQ4
IERESFBQUFQQDagECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhCxfYVFB1SEA8NCwoIBQQAQQAQFCAoLDQ8QEH1UFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAChBgYFBQQAeBAQIDAQwQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwcGAGIBAQICBgCHBxYKCQkJCAChBgUFBAMCAQEBAQIDAQwQFBQYGBwCPEQE
```



```

CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBAQEBAgYHBwcWCgkKCAgHBWYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRi
HBqADChI1DQoFAgEBAGMEBAoMEW8eTw4IVxkXCwkJBWYCOAIBAiIDAuUGBwgJCgICAQENAQEFaWI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUdAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBeQQiAgEBAf6RDAsLCQkICAYGBQUdAwIBAQIDAwUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBBQqNEakKCwvNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REaKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAAAAAAAAAABAAcAAQABAAAAAAAAACAACAABAAAAAAAAADAACADwABAAAAAAAAEAACAFgA
AAAAAAAAFAAsAHQABAAAAAAAAAGAAcAKAABAAAAAAAAAKACwALwABAAAAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAYBAGEDAQBBQEGAQCADXRYW5zcG9ydC12YW4LdXNlcil1b2RpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
content: "\e710";
}
.sf-icon-transport:before {
content: "\e702";
}
.sf-icon-payment:before {
content: "\e706";
}
.sf-icon-success:before {
content: "\e715";
}
}
</style>

```

**DISABLED.CS**

```

public ActionResult Demo()
{
 List<Step> iconOnly = new List<Step>();
}

```

```

iconOnly.Add(new Step { IconCss = "sf-icon-cart" });
iconOnly.Add(new Step { IconCss = "sf-icon-transport" });
iconOnly.Add(new Step { IconCss = "sf-icon-payment", Disabled = true });
iconOnly.Add(new Step { IconCss = "sf-icon-success" });
ViewBag.IconOnly = iconOnly;

return View();
}

```



### Setting active step

You can set the active step by specifying its index using the [ActiveStep](#) property. The default value is 0.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconOnly).ActiveStep(1).Render(
)
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAA
YBg9jYQhUBlAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAagAAAAAMAAAAUAMAAQAAABQABABKAAAAADAAIAAIABoCc5wbnCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAWADAAMAADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxcgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAyGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQCAQ0NDQwLCgoJCAGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgch/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHlwEjDwE1LwMjNz0BJZcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAqQHCPcBAGQGBgkKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfYQUExEQDw0LCggFBAEXBhCFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBg0HBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVfHcXfXyVFBIBSEA8NCwoIBQQAQQAoLDQ8QEHlUFRYXAAAAAQAQAAAAA/QDRwA/AH8

```

```

AhwCRAAABFR8OPw49AS8NkWEpDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICaKIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWQBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCagENDAwKCgghBQE
BAiKCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYKcQkJCACHBgUFBAMCAQEBAQIDAQWQBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwKRNRI
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAuUGBwgJCgICAQENAQEFaWI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQqIAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQWUBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwWGAgiwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAEEAACAFgA
BAAAAAAFAAAsAHQABAAAAAAGAACAABAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lvbiBNZXRybyBTdHVKaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQBByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAIAYQB0AGUAZAAGAHUAacwBpAG4AZwA
gAFMAeQBuAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAYBAGEDAQBBQEGAQcADXRYYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');

```

```

font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
content: "\e710";
}
.sf-icon-transport:before {
content: "\e702";
}
.sf-icon-payment:before {
content: "\e706";
}
.sf-icon-success:before {
content: "\e715";
}

```

```
}
</style>
```

### ACTIVESTEP.CS

```
public ActionResult Demo()
{
 List<Step> iconOnly = new List<Step>();
 iconOnly.Add(new Step { IconCss = "sf-icon-cart" });
 iconOnly.Add(new Step { IconCss = "sf-icon-transport" });
 iconOnly.Add(new Step { IconCss = "sf-icon-payment" });
 iconOnly.Add(new Step { IconCss = "sf-icon-success" });
 ViewBag.IconOnly = iconOnly;

 return View();
}
```



### Step status

Each step's progress state can be specified using the [Status](#) property. The possible values are **NotStarted**, **InProgress** and **Completed**. By default, the value is **NotStarted**.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
<div class="stepper-section">

@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).StepChanged("handles
tepChange").Render()
 <div id="paymentStatus" style="margin-top: 50px">Your payment has not
started yet</div>
</div>
<script>
 function handleStepChange() {
 var stepper = document.getElementById('stepper').ej2_instances[0];
 var stepStatus = stepper.steps[1].status;
 var statusMap = {
 'NotStarted': { text: 'Your payment has not started yet', color:
'#e74d4d' },
 'InProgress': { text: 'Processing your payment', color: 'orange'
},
 'Completed': { text: 'Payment successful', color: '#4CAF50' }
 };
 var currentStatus = document.getElementById("paymentStatus");
 if (currentStatus) {
 var { text, color } = statusMap[stepStatus];
 currentStatus.innerText = text;
 currentStatus.style.backgroundColor = color;
 }
 }
</script>
```

```

<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YBj9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYw1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AoAAAAACAACAAAAAABAAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAQAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAAAABAAAAQAQAAAAEAAAAAAGAAAAAUAAMAAQAABABKAAAADAAIAAIABoC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAawADAAMAAwADAAAAAEABAACMAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAgIBGcFBQQEAgH+CwEBAGQEBQUHBGgICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQCHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAChBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICACGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICACGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAy
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAQgGBGgKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKcAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDawMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBMQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKcAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQAQQAQCAoLDQ8QEhIUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECAGwQEBAYFBwYHCAgICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBGcHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAGwMEBQUGBGcCHCAcICQgICAcHBwYGBQUEBAICAQEBQICBAQFBQY
GBwCHCAgICQgHCAChBgYFBQQDAwIBAQIDAQwQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwcGAGIBAQICBgCHBxYKQKJCAChBgUFBAMCAQEBQIDAQwQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBQEBAGYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECAGwMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfXcWFxYXfHYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGgqYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwkRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAUGBwGJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAGwMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAYDIgICAQECaiI
CBAUGBwGJCQMBAGEMAQUDAwIDAQICBAQDBAQEBQEAwQEAQEBQICBAMFBQUGBwCICAgJBwgHBGc
GBgUFBAQEBQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQwUFBGyYICAKJCwsMKSckIiAeGxOYfhQ
TERAPDQLCgkIDxsJBQUFBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRCKCgUPFz0
REAKIBAMDawMCAQICAwcYAwEaBwQBAgIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAAAADAACADwABAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhcKRR

```

```

1ZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW51cmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGWAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAA
AAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlyY2sAAAA=) format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
#paymentStatus {
 padding: 10px;
 background-color: #e74d4d;
 color: white;
 border-radius: 5px;
 margin: 10px auto;
 text-align: center;
 font-weight: bold;
 max-width: 350px;
}
.stepper-section {
 width: 75%;
 margin: 0px auto;
 min-width: 85px;
 padding: 25px 0;
}
</style>

```

## STATUS.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
}

```

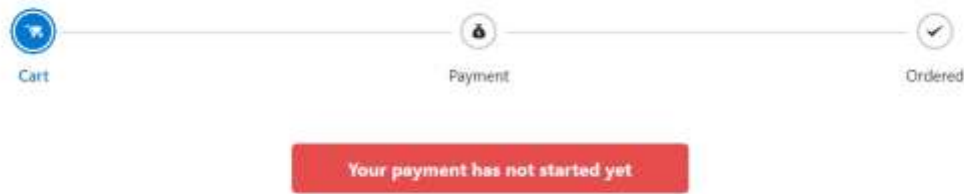
```

 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment" });
 iconLabel.Add(new Step { Label = "Ordered", IconCss = "sf-icon-success" });

 ViewBag.IconLabel = iconLabel;

 return View();
 }

```



### Step styling

You can use the [CssClass](#) property to customize the appearance of the each step.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAabwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAaZABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAAgAAAAAMAAAAUAMAAQAAABQABABKAAAADAAIAAIABOCc5wbncCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxBgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWefDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQwEAWMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQwEAgH+CwEBAGQEBAwMEBQUHBg
JCgkKCQoICQgHBwcFBQwEAWMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQwEAgH+CwEBAG
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAyGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAChBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgkJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgch/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHlwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIwAgQHCPcBAGQGBggKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECCBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHwYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHwYUFBMREQ4NDakIBgMBAQMGCakMDQ4RERMUFBYWFxcXfHwY

```

```
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCBlwICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEAS8PDg0MCwoKCAYGAWMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBiSEA8NCwoIBQQBAQQFCAoLDQ8QEHuIFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECaWQEBAYFBwYHCAgICAgICaIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICaKIBwgHBwYGBQUEAwMCAQECaWMEBQUGBgCHCAcICQgICaCHBwYGBQUEBAICAQEBaQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQWFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgghBQE
BAikCAGIEAwQFDA0SBwcGAGIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBaQIDAQWFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBaQEBaGYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECaWMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHfYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBaWkRNRI
HBqADChI1DQoFAGeBAgMEBAoMEw8eTw4IVxkXCwkJBWYCOAIBaIIDAUGBwgJCgICAQENAQEFaWI
DAgECAGMFAwMEBAUDBAMFAwIBAQECaWMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECaIi
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBaQEAwQEAQEBaQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBQqIAGeBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQWFBgYICaKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAGIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDGcIJCYNmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAAcACAABAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZlZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctcgU3luY2Z
1c2lvbiBNZXRybyBTDHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYABwBuAHQAIAABnAGUAAbgB1AHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaAaA
AAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2h1Y2sAAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
```



```

 content: "\e706";
 }
 .sf-icon-success:before {
 content: "\e715";
 }

 /* Custom styles */
 #stepper .custom-step .e-step-label-optional {
 font-style: italic;
 font-weight: 900;
 color: #299100;
 }
</style>

```

### CSSCLASS.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart" IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address" IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment" IconCss = "sf-icon-payment",
 CssClass = "custom-step", Optional = true});
 iconLabel.Add(new Step { Label = "Confirmation" IconCss = "sf-icon-success" });
 ViewBag.IconLabel = iconLabel;

 return View();
}

```



### Step validation

You can set the validation state for each step to displaying a success or error icon by using [IsValid](#) property.

To know more about Stepper validation, refer the [Validation](#) section.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("iconStepper").Steps(ViewBag.IconSteps).Render()
@Html.EJS().Stepper("labelStepper").Steps(ViewBag.LabelSteps).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAIAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYbG9jYQhUBlAAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM

```

AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABgABAAAAQAARxT6wV8  
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAEEAAAAGAR8ABgAAAAA  
AAgAAAAoACgAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA  
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA  
EAAAABAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCOc  
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAaWADAAMAaWADAAAAAEABAACAAMABQAAAAA  
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8  
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk  
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBggICAK  
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQcHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw  
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY  
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQ4LcwsJCQkHBwUFaWKE/eMBAQoJCQkJCAcHBgYFBAM  
DAQEBAQMDBAUGBgHCAkKCgkJCQgICAcGBgQFAwICAgIDBAUFBgHCAkKCgkJCgkJCgkJCgkJCg  
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBWYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI  
GehYJBKYp/l0ICAcGBQQAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo  
BAGQFBgC/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY  
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHlWejDwe1LwMjNz0BJzcfBTcfAg8BLwY  
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ  
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY  
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAGQGBgKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY  
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak  
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXfHQUEXEQDw0LCgGfBAEXBhCFBAMDrxYWDQEBawUHCAsmDQ4  
IERESFBQUFQDAGECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ  
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA  
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0  
PEBESFBUVFhCXfYVFBISEA8NCwoIBQQAQQFCAoLDQ8QEHlUFRYXAAAAAQAaaaaa/QDRwA/AH8  
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM  
EBAQGBQcGBwGICAgICAgIBWYHBQYEBQDAgEBAgMEBAQGBQcGBwGICAgICAgIBWYHBQYEBQDAgH  
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBgQEBAMCAQECAGQEBAYFBWYHCAgICAgICAcIBGcFBgQ  
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBGUFBAQCAgEBAQECAGQEBQUGBGcHBwG  
ICAKIBwGHBWYGBQEAwMCAQECAGMEBQUGBGcHCAcICQgICAcHBWYGBQUEBAICAgEBAQICBAQFBQY  
GBwCHCAgICQgHCAcHBGYFBQQDAWIBAQIDAQWQFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAADAAA  
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8  
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCgGHBQE  
BAIkCAGIEAwQFDA0SBwCGAgIBAQICBgCHBxYKQKJCAcHBGUFBAQCAQEBQIDAQWQFBQYGBwCPEQE  
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBQEBAGYHBwCWCgkKCAgHBWYFBQQDAgE  
BAQECAGMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHYWFhYVFRUVFAQCAQI  
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGgqYGCEsZDQ0NDA0MCwsKCQgIBGUEAgIBaQQAQEBAwkRNRI  
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBWYCOAIBaIIDAQWUGBwGJCgICAQENAEFAwI  
DAGECAGMFAwMEBAUDBAMFAWIBAQECAGMEBAUGBgYHCAgICQgHBwCGBGUFBAQEBAYDIgICAQECaI  
CBAUGBwGJCQMBAGEMAQUDAwIDAQICBAQDBAQEBQEAwQEAQEBQICBAMFBQUGBwCICAgJBwGHBGc  
GBGUFBAQEBQqIAGEBaf6RDAsLCQkICAYGBQUdAwIBAQIDAQWUFBgYICAKJCwsMKSckIiAeGxoYfHq  
TERAPDQwLCgkIDxsJBQUBQqNEAKKCwwNDxAREXQWGBobHiAiJCcCoAMDawQECA8XPRcKCGUPFz0  
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs  
EJwGAGIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM  
BARQuNGsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAA  
AAAAAAEAAAABAAAAAABAACAAQABAAAAAACAACAABAAAAAADAACADWABAAAAAAEAACAFgA  
BAAAAAAEFaAsAHQABAAAAAAGAACAKAABAAAAAAKACwALWABAAAAAALABIAWwADAAEECQAAAAI  
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA  
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAYwADAAEECQALACQBIyBEZWZhdWx0UmVndWxhcR  
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z  
lc2lvbiBNZXRybyBTdHVKaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB  
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQBByAHMAaQBvAG4AIAAxA  
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbGBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA  
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM  
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa

```

AAAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlci1tb2RpZnkRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlyY2sAAAA=) format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
#labelStepper {
 margin-top: 50px;
}
</style>

```

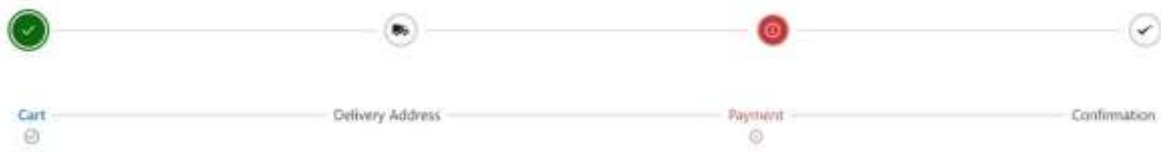
## VALIDATION.CS

```

public ActionResult Demo()
{
 List<Step> iconSteps = new List<Step>();
 iconSteps.Add(new Step { IconCss = "sf-icon-cart", IsValid = true });
 iconSteps.Add(new Step { IconCss = "sf-icon-transport" });
 iconSteps.Add(new Step { IconCss = "sf-icon-payment", IsValid = false });
 iconSteps.Add(new Step { IconCss = "sf-icon-success" });
 List<Step> labelSteps = new List<Step>();
 labelSteps.Add(new Step { Label = "Cart", IsValid = true });
 labelSteps.Add(new Step { Label = "Delivery Address" });
 labelSteps.Add(new Step { Label = "Payment", IsValid = false });
 labelSteps.Add(new Step { Label = "Confirmation" });
 ViewBag.IconSteps = iconSteps;
 ViewBag.LabelSteps = labelSteps;

 return View();
}

```



## Step types in ASP.NET MVC Stepper control

The Stepper control provides support for displaying steps with the following step types.

### Default type

In default type, the Stepper displays steps with a combination of both indicators and labels by setting the [StepType](#) property as [Default](#). By default, the Stepper displays steps in the [Default](#) type.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).StepType(StepType.Default).Render()

<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YBg9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1lUf5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAAAAABAAAAAQAAAA
AAAAAAAAAAAAAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAIAAIABOcC5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAAABAAwADAAMAAwADAAAAAEABAACAAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAgIBGcFBQQEAgH+CwEBAGQEBQUHBGgICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQCHBwGJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAChBgYFBAM
DAQEBAQMDBAUGBgCHCAkKCgkJCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkKCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkKCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwGJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQgQHCPcBAgQGBgGKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEAP8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXfHYUEXEQDw0LCgGFBAEXBhCFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUFQDAgECAGMEBAUGBgYIBwGJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgohBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEAP8PDg0MCwoKCAyGAWMBAQMDBgYICgoLDA00Dw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhCXfYVFBISEA8NCwoIBQQAQQFCAoLDQ8QEHlUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAGM
EBAQGBQcGBwGICAgICAgIBwYHBQYEBQDAgEBAgMEBAQGBQcGBwGICAgICAgIBwYHBQYEBQDAgH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBgQEBAMCAQECAGQEBAYFBwYHCAgICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwG
```

```
ICAkIBwgHBWYGBQUEAwMCAQECaWMEBQUGBgcHCAcICQgICAChBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAIkCAgIEAwQFDA0SBwcGAgIBAQICBgcHBxYKCQkJCAcHBgUFBAMCAQEBAQIDAQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBAQEBAgYHBwcWCgkKCAgHBWYFBQQDAgE
BAQECAWMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBAwKRNRI
HBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJBWYCOAIBAiIDAUGBwgJCgICAQENAEFAWI
DAgECAGMFAWMEBAUDBAMFAWIBAQECAWMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAyDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAQWUFBgYICakJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQqNEAkKCwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRCKCgUPFz0
REAkIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsKqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACACAABAAAAAADAACADwABAAAAAEEAACAFgA
BAAAAAAFAASAHQABAAAAAAGAACAABAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArAB1AGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAAQBVAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAAQBVAG4AIAABNAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAAQBVAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAA
AAAAAAYBAGEDAQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkc2hvcHBpbmctY2F
ydfF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
content: "\e710";
}
.sf-icon-transport:before {
content: "\e702";
}
.sf-icon-payment:before {
content: "\e706";
}
.sf-icon-success:before {
content: "\e715";
}
}
</style>
```

**DEFAULT.CS**

```
public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address", IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment" });
 iconLabel.Add(new Step { Label = "Confirmation", IconCss = "sf-icon-success" });

 ViewBag.IconLabel = iconLabel;

 return View();
}
```

**Label type**

In label type, the Stepper displays the steps with only the step labels by setting the [StepType](#) property as [Label](#).

When both label and text are defined, the label takes priority in displaying the steps.

**CSHTML**

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).StepType(StepType.Label).Render()
```

**LABEL.CS**

```
public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address", IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment" });
 iconLabel.Add(new Step { Label = "Confirmation", IconCss = "sf-icon-success" });

 ViewBag.IconLabel = iconLabel;

 return View();
}
```



### Label positions

You can display the label on the top, bottom, left, or right side of the steps using the [LabelPosition](#) property.

The following label positions are supported in Stepper:

- | Value         | Description                                         |
|---------------|-----------------------------------------------------|
| ----- -----   |                                                     |
| <b>Top</b>    | Positions the label at the top of each step.        |
| <b>Bottom</b> | Positions the label at the bottom of each step.     |
| <b>Start</b>  | Positions the label to the left side of each step.  |
| <b>End</b>    | Positions the label to the right side of each step. |

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).LabelPosition(StepLabelPosition.Top).Render()

<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr+pwAAAggAAAjQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAAAYbG9jYQhUblAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABgABAAAAQAARxT6wV8PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAEEAAAGAR8ABgAAAAAAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAaokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMMAAAA
 AAAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
 EAAAAAABAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAAADAAIAAIABoCc5wbnCOc
 Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAA
 AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxBgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
 NAR8FFSM1JxEfBz8OOWEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
 IBwchBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
 JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
 D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
 HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLcwsJCQkHBwUFAwKE/eMBAQoJCQkJCAChBgYFBAM
 DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAgGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwC
 GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
 GehYJBKYp/10ICAgGBQQCAQ0NDQwLcgoJCAGGBQUdAgIDBQUgCAGJCgoLDA0NDQECBAUGBwQI1fo
 BAgQFBgch/iwNDawLcwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
 AQgBaAGwArQDuAAABBzcfAwUhLwIH1y8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
 3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
 CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY
 GBxwCAwIBFQYGBAgFBgIWAQgHCPcBAgQGBgkKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
 GCAkLCwwNDg8PEBAQEa8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
 IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhCFBAMDrxYWDQEBawUHCAsMDQ4
 IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
```

```

qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwWNDg8PEBA
QEA8PDg0MCwoKCAYGAWMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEHuIFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBWgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBWgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwGICAgICAgHBGcFBgQEBAMCAQECaWQEBAYFBwYHCAGICAgICaIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwGJZ/dzDAf8AAQkICAgHBWcGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICaKIBWgHBWYGBQUEAwMCAQECaWMEBQUGBgCHCaICQgICaCHBwYGBQUEBAICAQEBaQICBAQFBQY
GBWcHCAgICQgHCAcHBgYFBQQDAWIBAQIDAQWFBQYGBWcIBWgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCgghBQE
BAikCAgIEAwQFDA0SBWcGAgIBAQICBgCHBxYKCQkJCAcHBgUFBAMCAQEBaQIDAQWFBQYGBWcPEQE
CAhUCAgINDAsLCQgHBQICKQICAgQDBAULDhIHBWYCAQEBaQEBaYHBWcWCgkKCAgHBWYFBQQDAgE
BAQECaWMEBAYFBgCHBABAQED/qwUFRUVFRYWFhYWFxYXfHcXFxcWFxYXfHhYWFhYVFRUVFAQCAQI
CBAUGCagJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBaWkRNRI
HBqADChI1DQoFAGeBAgMEBAoMEw8eTw4IVxkXCWkJBwYCOAIBaIIDAuUGBwgJcGICAQENAEFAWI
DAgECAGMFAWMEBAUDBAMFAWIBAQECaWMEBAUGBgYHCAGICQgHBWcGBgYFBQQEBAYDIgICAQECaI
CBAUGBwgJcQMBAGEMAQUDAWIDAQICBAQDBAQEBaQEAwQEAQEBaQICBAMFBQUGBwCICAgJBWgHBGc
GBgUFBAQEBQqIAGeBAf6RDAsLCQkICAYGBQUDAWIBAQIDAQWFBgYICaKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAKCwWNDxARExQWGBobHiAiJCcCoAMDawQECAXPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAGIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDGcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAAcACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lvbiBNZXRybyBtdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAacwBpAG4AZwA
gAFMAEQBuAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuaGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaAa
AAAAAAYBAGEDAQQBBQEGAQcADXRyYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}

```



```

 }
 .sf-icon-success:before {
 content: "\e715";
 }
</style>

```

### LABELPOSITION.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address", IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment" });
 iconLabel.Add(new Step { Label = "Confirmation", IconCss = "sf-icon-success" });

 ViewBag.IconLabel = iconLabel;

 return View();
}

```



### Indicator type

In indicator type, the Stepper displays steps with only the step indicators by setting the [StepType](#) property as `Indicator`.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.StepperSteps).StepType(StepType.Indicator).Render()
@Html.EJS().Stepper("textStepper").Steps(ViewBag.TextOnly).StepType(StepType.Indicator).Render()

```

### INDICATOR.CS

```

public ActionResult Demo()
{
 List<Step> textOnly = new List<Step>();
 textOnly.Add(new Step { Text = "1" });
 textOnly.Add(new Step { Text = "2" });
 textOnly.Add(new Step { Text = "3" });
 textOnly.Add(new Step { Text = "4" });
 List<Step> defaultSteps = new List<Step>();
 defaultSteps.Add(new Step { });
 defaultSteps.Add(new Step { });
 defaultSteps.Add(new Step { });
}

```

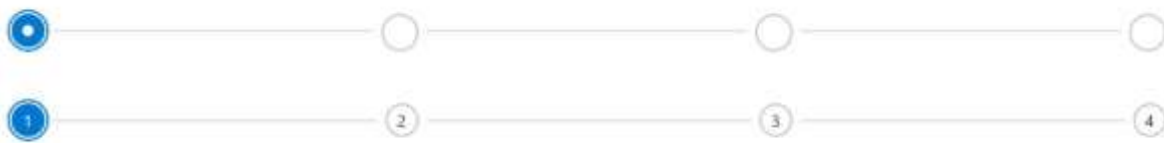
```

defaultSteps.Add(new Step { });

ViewBag.TextOnly = textOnly;
ViewBag.StepperSteps = defaultSteps;

return View();
}

```



## Orientation in ASP.NET MVC Stepper control

The Stepper control supports the display of steps in both horizontal and vertical orientations by using the [Orientation](#) property.

### Horizontal

In horizontal orientation, the steps are displayed in a side-by-side manner by setting the [Orientation](#) property to **Horizontal**. By default, the steps are displayed in the horizontal orientation.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconOnly).Orientation(StepperOrientation.Horizontal).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDIPaAAABmAAAAF5nbHlmEwr+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAAAYAAAAAYbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoMAAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAAAGAAAAoACgAAAP8AAAAAAAAAAQQAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOcc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWefDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAgGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/l0ICAgCBQQCAQ0NDQwLCgoJCAGGBQUdAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgCH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJZcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAg8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQgQHCPcBAgQGBgkKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECEBAY

```

```

GCAkLCwWNDg8PEBAQEAS8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDAk
IBgMBAQMGCAkMDQ4RERUMFUYWFxcXfHQUExEQDw0LCggFBAEXBhCFBAMDrxYWDQEBaWUHCAsmDQ4
IERESFBQUFQQDAGECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwWNDg8PEBA
QEAS8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEHuIFRYXAAAAAQAaaaaa/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBWYHBQYEBaQDAGEBAGMEBAQGBQcGBwgICAgICAgIBWYHBQYEBaQDAGH
+aAICAgQEBAYFBWYIBwgICAgICAgHBGcFBgQEBAMCAQECaWQEBAYFBWYHCAgICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICAkIBwgHBWYGBQUEAwMCAQECaWMEBQUGBgCHCAcICQgICAcHBWYGBQUEBAICAQEBaQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAWIBAQIDAQWFBQYGBWcIBWgB+wH+vQFABP5dOgGkAEAAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDW0dAQ8BKwIvAT0BLwc9AT8COWefBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCggHBQE
BAIkCAGIEAwQFDA0SBwcGAgIBAQICBgCHBxYKQKjCACHBgUFBAMCAQEBaQIDAQWFBQYGBWcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBaQEBAGYHBwcWCgkKCAgHBWYFBQQDAGe
BAQECaWMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBaWkRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCWkJBwYCOAIBaIIDAQWUGBwgJCgICAQENAQEFaWl
DAGECAGMFAWMEBAUDBAMFAWIBAQECaWMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECaIi
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBaQEAwQEAQEBaQICBAMFBQUGBwcICAgJBwgHBGc
GBgUFBAQEBQqIAGEBaf6RDAsLCQkICAYGBQUDAwIBAQIDAQWFBgYICAKJCwsMKSckIiAeGxoYfHq
TERAPDQwLCgkIDxsJBQUFBQqNEAKKCwWNDxARExQWGBobHiAiJCcCoAMDawQECa8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwWGAgiWxmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNGsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAAABAAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAACADWABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEEA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhcR
lZmFlbHREZwZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lvbiBNZXRybyBTDHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgB1AHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbWw3AHcAdwAuAHM
AeQBuAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAA
AAAAAAYBAGEDAQQBBQEGAQcADXRYW5zcG9ydC12YW4LdXNlcil1b2Rpb2RpbmctY2FyY2F8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
content: "\e710";
}
.sf-icon-transport:before {
content: "\e702";
}

```

```
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
</style>
```

**HORIZONTAL.CS**

```
public ActionResult Demo()
{
 List<Step> iconOnly = new List<Step>();
 iconOnly.Add(new Step { IconCss = "sf-icon-cart" });
 iconOnly.Add(new Step { IconCss = "sf-icon-transport" });
 iconOnly.Add(new Step { IconCss = "sf-icon-payment" });
 iconOnly.Add(new Step { IconCss = "sf-icon-success" });
 ViewBag.IconOnly = iconOnly;

 return View();
}
```



## Vertical

You can display the steps one below the other vertically by setting the **Orientation** property to Vertical.

## CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconOnly).Orientation(StepperOr
ientation.Vertical).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXcDeIPaAAABmAAAAF5nbHlmeWr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIuUQQAHAHAARAAACRobXR4GAAAAAAYAAAAA
YbG9jYQhUBIAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAABAAAAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAAMAAAUAMAAQAAABQABABKAAAAADAAIAAIABOCc5wbnCOc
Q5xx//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAIwC3AQkBgGAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSMlJxEfBz8OOWEfDjM/BzUnIw8GATM/Dx8PMxehAQ8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAGIBgcFBQQEAgH+CwEBAgQEGBQUHBggICAK
ICgCkQwQoICQgHBwCfBQQDAwEBAQEDAwQFBQcHBwgJCAoJCGoKCQkICAgGBwUfBAQCAQJ8AwUIWAw
D3n0BAwMGBqYICAMEBOYHBwkJCAsLDA0ND04OD04MDAwLCakJCAYGBQMDKAQIBwYFBAECvLsICAY
```

```

HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkKCgoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAgQFBgCH/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAABAgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHIy8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAqQHCPcBAgQGBgKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEa8PDg0MCwoKcAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXFhYUFBMREQ4NDak
IBgMBAQMGCAkMDQ4RERMUFBYWFxcXFhQUExEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsMDQ4
IERESFBQUBFQDAGECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAqIGBQMCAQQDBgZ
qBg0HBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEa8PDg0MCwoKcAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXFxYVFBISEA8NCwoIBQQAQQFCAoLDQ8QEhIUFRYXAAAAAQAaaaaa/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECaWQEBAYFBwYHCAgICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBgCHBwg
ICakIBwgHBwYGBQUEAwMCAQECaWMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAqIDAQFQBQYGBwCIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAGENDAwKCggHBQE
BAikCAGIEAwQFDA0SBwcGAGIBAqICBgCHBxYKQKkJCAcHBgUFBAMCAQEBAQIDAQFQBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBAQEBAgYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECaWMEBAYFBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXFhcXFxcWFxYXFhYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwKRNRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAWMEBAUGBgYHCAgICQgHBwCGBgYFBQQEBAyDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDDBAQEBAQEAWQEAQEBAGICBAMFBQUGBwCIBAgJBwgHBGc
GBgUFBAGQBQqIAgEBAf6RDAsLCQkICAYGBQUDAwIBAqIDAQFQBQYGBwCIBwgYICakJCwsMKsckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAkKCwwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRCKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIiJkIAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAsAHQABAAAAAAGAAcAKAABAAAAAAAKACwALwABAAAAAALABIawwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBiyBEZWZhdWx0UmVndWxhcR
lZmFlbHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
1c2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGUAZAAGAHUAacwBpAG4AZwA
gAFMAeQBuaGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuaGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
AAAAAAYBAGEDAQBBQEGAQcADXRYyW5zcG9ydC12YW4LdXNlcil1tb2RpZnkc2hvcHBpbmctY2F
ydf8wMS0LC3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueype');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;

```

```
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
}
</style>
```

### VERTICAL.CS

```
public ActionResult Demo()
{
 List<Step> iconOnly = new List<Step>();
 iconOnly.Add(new Step { IconCss = "sf-icon-cart" });
 iconOnly.Add(new Step { IconCss = "sf-icon-transport" });
 iconOnly.Add(new Step { IconCss = "sf-icon-payment" });
 iconOnly.Add(new Step { IconCss = "sf-icon-success" });
 ViewBag.IconOnly = iconOnly;

 return View();
}
```



### Linear flow in ASP.NET MVC Stepper control

The Stepper control enables users to progress sequentially through each step, ensuring navigation from one step to the next in a linear way by setting the [Linear](#) property to `true`. The default value is `false` allowing navigation between any steps and vice versa.

The example demonstrates the functionality of both linear and non-linear flow in the Stepper.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).Linear(true).Render(
)
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMj1vSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
```

```

AAgAAAAoACgAAAP8AAAAAAAAAAQQA ZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQM AAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAaGAAAAAMAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCOc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAAEABAACAAMABQAAAAA
AAAEQAiwc3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKWEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkJCAGIBGcFBQQEAgH+CwEBAGQEBQUHBGgICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQcHBWgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwCICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUWQI1fo
BAGQFBGcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABBzcfAwUhLwIHlY8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBAwMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQHCPCBAgQGBGgKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhCFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQDDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEHlUFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKWEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBWgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBWgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECawQEBAYFBwYHCAGICAgICAcIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBgUFBAQCAgEBAQECAGQEBQUGBGcHBWg
ICAkIBwgHBwYGBQUEAwMCAQECawMEBQUGBGcHCAcICQgICAcHBwYGBQUEBAICAQEBQICBAQFBQY
GBwCHCAgICQgHCAHBGyYFBQQDAwIBAQIDAwQFBQYGBwCIBwgB+wH+vQFABP5dOgGkAqEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIY8HDWYdAR8WDw0dAQ8BKwIvAT0BLwC9AT8COWefBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAgENDAwKCgHBQE
BAIkCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYKQKJCAcHBGUFBAWCAQEBQIDAQWQFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULDhIHBwYCAQEBQEBAGYHBwCWCgkKCAgHBwYFBQQDAgE
BAQECawMEBAYFBGcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHYWFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQQAQEBawKRNRI
HBqADChI1DQoFAGeBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECawMEBAUGBgYHCAGICQgHBwCGBgYFBQQEBAYDIgICAQECaiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBQEAwQEAQEBQICBAMFBQUGBwCICAgJBwgHBGc
GBgUFBAQEBQQiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAUFBgYICAKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUFBQqNEAKKCwwNDxARExQWGBobHiAiJCcCoAMDawQECAXPRcKCGUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwGAGIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNGsMDGcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAACAACAABAAAAAADAACADwABAAAAAAEAACAFgA
BAAAAAAFAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIawADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhcR
lZmFlbHREZWNhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctU3luY2Z
1c2lubiBNZXRybyBTDHVkaW93d3cuc3luY2Z1c2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAyQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgB1AHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
AAAAAAYBAgEDAQQBBQEGAQCADXRYW5zcG9ydC12YW4LdXNlcil1b2Rpb2R2hvcHBpYmctY2F
ydf8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
font-weight: normal;

```



```

 font-style: normal;
 }
 [class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 }
 .sf-icon-cart:before {
 content: "\e710";
 }
 .sf-icon-transport:before {
 content: "\e702";
 }
 .sf-icon-payment:before {
 content: "\e706";
 }
 .sf-icon-success:before {
 content: "\e715";
 }
}
</style>

```

### LINEAR.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address", IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment" });
 iconLabel.Add(new Step { Label = "Confirmation", IconCss = "sf-icon-success" });

 ViewBag.IconLabel = iconLabel;

 return View();
}

```

### Steps validation in ASP.NET MVC Stepper control

The Stepper control allows you to set the validation state for each step, displaying either a success or error icon. You can define the success state of a step by setting the [IsValid](#) property to `true`. If set to `false`, the step will display an error state. By default, the [IsValid](#) property is `null`.

Based on the [StepType](#), the validation state icon will be displayed either as an indicator or as part of the step label/text.

### CSHTML

3342

```

BAAAAAAFAAAsAHQABAAAAAAGAAcAKAABAAAAAAKACwAlWABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmF1bHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lvbiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAG4AIAABnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAG4AIAABnAGUAdABYAG8AIABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AAAAAAYBAGEDAQQBBQEGAQcADXRYYW5zcG9ydC12YW4LdXNlcil1tb2RpZnkRc2hvcHBpbmctY2F
ydfF8wMS0Lc3BlbmQtbW9uZkFY2hly2sAAAA=) format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
#labelStepper {
 margin-top: 50px;
}
</style>

```

## VALIDATION.CS

```

public ActionResult Demo()
{
 List<Step> iconSteps = new List<Step>();
 iconSteps.Add(new Step { IconCss = "sf-icon-cart", IsValid = true });
 iconSteps.Add(new Step { IconCss = "sf-icon-transport" });
 iconSteps.Add(new Step { IconCss = "sf-icon-payment", IsValid = false });
 iconSteps.Add(new Step { IconCss = "sf-icon-success" });
 List<Step> labelSteps = new List<Step>();
 labelSteps.Add(new Step { Label = "Cart", IsValid = true });
 labelSteps.Add(new Step { Label = "Delivery Address" });
 labelSteps.Add(new Step { Label = "Payment", IsValid = false });
 labelSteps.Add(new Step { Label = "Confirmation" });
}

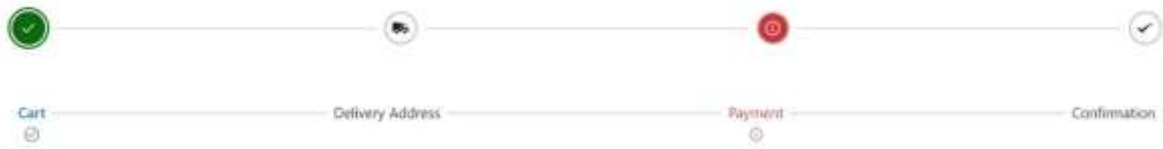
```

```

ViewBag.IconSteps = iconSteps;
ViewBag.LabelSteps = labelSteps;

return View();
}

```



### Template in ASP.NET MVC Stepper control

The Stepper control allows you to customize the default appearance and content of each step, creating a personalized experience for the user. You can use the [Template](#) property to set the template content for the steps.

The [StepModel](#) and current step index are passed as **Step** and **CurrentStep** properties in the template context for customization.

#### CSHTML

```

@using Syncfusion.EJ2.Navigations
<div class="default-stepper-section">

@Html.EJS().Stepper("stepperTemplate").Steps(ViewBag.IconLabel).ActiveStep(1)
.Template("#template").Render()
</div>
<script id="template" type="text/x-jsrender">
 <div class="template-content">

 ${step.label}
 </div>
</script>
<style>
 .template-content {
 background: #fff;
 width: 65px;
 }
 /* Stepper progressbar customization */
 .e-stepper .e-stepper-progressbar {
 height: 3px;
 top: 25px;
 }
 .e-stepper .e-stepper-progressbar .e-progressbar-value {
 background-color: #27d96d;
 }
 /* Stepper status customization */
 #stepperTemplate .e-step-completed * {
 color: #19cd60;
 }
 #stepperTemplate .e-step-inprogress * {
 color: #3479f3;
 }
</style>

```

```

#stepperTemplate .e-step-notstarted * {
 color: #bdbdbd;
}
@@font-face {
 font-family: 'template_updated';
 src:
 url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1tSfUAAAEoAAAAVmNtYXDNdE+dkAAABlAAADxnbHlm39z
zMQAAAdwAAAacaGVhZCaImHMAAADQAAAAANmhoZWEIUQQGAAAArAAAACRobXR4FAAAAAAAYAAAA
UbG9jYQR8BVAAAAHQAAADG1heHABFwEbAAABCAAAACBuYW1ldkXdggAACHgAAAKRcG9zdD2fuhI
AAAsMAAAAXwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABQABAAAAAQAApDfGf18
PPPUACwQAAAAAOG7KcEAAAAA4bspwQAAAAAD9AP0AAAAACAACAAAAAABAAAAAFAQ8ACAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIAABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAwQAAAAAXAQAAAAAABAAAAAABAAAAAQAQAAAA
EAAAABAAAAQAQAAAAAACAAAAwAAABQAAwABAAAFAAAEACgAAAAEAAQAAQAA5wP//wAA5wD//wA
AAAEABAAAAEAAgADAAQAAAAAAVQCAGMoA04ACAAAAAAD0wP0AB8APABcAHwA+gD+AQoBDgAAAQ8
HLWY9AT8GHwY1DwcvBj0BPwI7AR8FNw8HLWY9AT8GHwY1DwYrAS8GPwczHwUHFR8HMzcXDwIfBz8
HJzcfAz8CFw8BHwc/By8HDwMnPW1lLwcPBxUfAQcvBCMPAic/Ay8HDwY1ESERAYEHFzCzFzcnIRE
hJyE1IQMiAQECAwQEBQUFBAUDAwICAgIDAUEBQUFBAQDAgH+mwEBAGMEBAUFBQQFAwMCAgILBgU
FBQQEAWIBnAEBAGMEBAUFBQQEBAMCAgICAwQEBUFBAQDAgEAWIB/rkBAQMDAwUEBQUFBAQDAgEBAQE
CAwQEBQUFBAUDAwMBUAEDBgYJCQsLCAXoAgQBAQFwBgJCwsLCwkJBgYDAQFECAGICAKIBIEBAQE
DBQCICGoLDAoJCQCFAwEBAwUHCQkKDAoJCQ14BQMCAQMFwBgKCgwLCGoIBwUDAQICPwEJCQoMBwc
GCGQFAWIBAQQFwBgJCwsLCwkJBgUEAsb89j8Bz8sr+gX7K8sBZ/x4DwOm/FoB7QUFBAQDAgEBAQE
CAwQEBQUFBAUDAwMBAQEBAwMDBQZzBQUEBAMCAQEBAQIDBAQFBUQCwICAgMEBARgBQUEAwMDAQE
BAQMDAwQFBUFBAMDAwEBAQEDAwMEBUfBAQEAWICAgIDBAQEQUFBAQDAgEBAQIDBAQFBUQCgo
IBwUDAQNNBASLCwsJCQYFBAEBAwYGCQkLCWkuBAMCAQECAVkfCwsLCQgHBQQAQFwBgJCwsLCwk
JBgYDAQECBQZTCACJCAsLCQgHBQQAQFwBgJCwsJCAUqAgcFAwEBAGRkCQgICAwKCgGHBQMBAQM
FBwGKcMp97gIS/bDALE3tLcACji8+AAAGAAAAAP0A+QAEWA5AesAdwCFAIkAAAEzPwc1LwcjFxU
PDisBFsMRNx8OBsM1Mx8Njz8CFTMVDw4vAxUzFSMVMxUjFSE/AhEvAiE1IREzPwMRLwMhJREFEQE
QIgcHBgUFBAECAQIEBQUGBAYmiAECAGYEBQYGCACJCakKCQokNlSMCwoKCQgHBwYFBAMDAgEBgn0
QDAwLCwoJCQgHBgUEAwK7BA4NnAIEBAHYHCAkJCgsMDAwNDQcWDgza2traAU0FAwICAwX+swf3FQQ
DAGEBAGMC/nL9rgIyAgABAgQFBgYDBYQHBgYFBQQBAIEfCQoJCQkIBwcGBgUEAwICCAFIBQEBAgM
DBQUGBgICAKKCgV9AgMEBgYHCAkJCwsLDA1QAgUBhA0NDAsKCgoIBwcGBQQCAGEBAGMELSA+Hz8
CBAQB/wUDAiD+SgECAwIBwwQDAgFb/PdfA8gAAAAACAAAAAPzAyKAcQEIAAABDyAVHw41Pw01Lwo
1LxErAQ8CKwEvCQcnDwgvBSsCDxQVDw4VHw8zLwQ1Px8fCj8ELxMPAgICCwwLCgoKCQkLCGgIBgU
EAWECEg8QDg0LCgkIBgUEAgEBAQIEBQYICQoLDQ4PDg4PAh0ODgYHBgCHBQUFAwMDBAYDBgcICQo
MDBMCAgQCBAQFBgYGBwCHCAgJCRIUEwoSEhQCARYICw0NDQ4PDw8XFwoVExIQDw4NCQwLCwsLCws
LCwoLCwoLCgsJCgGHBwYGBQQDAwIBARAPDgWLCgkHBwYEBAMBAQECAUGBwGJCgoKCgoLCwtgCwY
FAwICAwQGBwKcKw0NDg8QEgIBBAMHBwGKCwwNDxITExMUFBUDw8ODQwMCwsJCAgJERsFBAQFBgY
HCAgJCgoLDAwNFBuUFhELCwKEAgMEBQUHBwGNDQ0PDw8QEgIBAgMFBgYICAKKCwwOCgsKCgsKFA8
NDAsKCQCGBAMBAQEBBAMDBAYGBwCICAKIGBIIdCA0LCgGHBgUGAhAREQkICAKICACHBgYFBAQDBQM
DBQgbBwoJCAYFAwIBAZ0DBwkLDQ4QEg8GBAQDAgICAgQEBQYGCACICQkJCgkLCgsLDAsZAwQGBgc
ICAKKCwsLDAwMDA0NDQsMCgoKCQCGBQQDAWEUEBAQEBEQE8PDQwLCgkJBwYFBQECDgWREA8ODgw
LCwoIBgUCAQEDBAUGBwGJCwsMBAMDAWEZDQ0NDAsLCwoJCQgICAYGCAYEAgEBAgADAAAAAAPfA/Q
ACwAPABMAAAEHFzCvMzUXNyc1IyUHESEnITUhAdq+NY1LiJw/S/6JAzX8y0MDwPxAAQC/NYp+foo
1v1ErAc5DZwAAAAASAN4AAQAAAAAABAAAAAQAQAAEAAQAAAAAAGAHABEAAQAAAA
AAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgAQAEMAAQAAAAACgAsAFMAAQ
AAAAQcASAH8AAwABBAkAAAAACAJEAwABBAkAAQAgAJMAwABBAkAAgAOLMAAwABBAkAAwAgAME
AAwABBAkABAAgAOEAwABBAkABQAWAQEAwABBAkABgAgARcAAwABBAkACgBYATcAAwABBAkACwA
kAY8gdGVtcGxhdGVfdXBkYXRlZFJlZ3VsYXJ0ZW1wbG90ZGV0ZV91cGRhdGVkdGVtcGxhdGVfdXBkYXR
lZFJlcnNpb24gMS4wdGVtcGxhdGVfdXBkYXRlZEZvbncGZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAdABlAG0AcABsAGEAdABlAF8AdQB
wAGQAYQB0AGUAZABSAGUAZwB1AGwAYQByAHQAZQBtAHAABABhAHQAZQBfAHUAACABkAGEAdABlAGQ
AdABlAG0AcABsAGEAdABlAF8AdQBwAGQAYQB0AGUAZABWAGUAAGBzAGkAbwBuACAAMQAuADAAdAB
lAG0AcABsAGEAdABlAF8AdQBwAGQAYQB0AGUAZABGAG8AbgB0ACAAZwB1AG4AZQByAGEAdABlAGQ
AIABlAHMAAQBuAGcAIABTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQACgBvACAAUwB0AHUAZAB
pAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAGAAAAAaKAAAAAA

```

```

AAAAAAAAAAAAAAAAAAAAAAAAFAQIBAwEEAQUBBgANcHJvamVjdG9yLW9sZApwb3dlcnBvaW50CG9
uZWRYaXZlDXByb2ply3RvciluZXcAAAA=) format('truetype');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'template_updated' !important;
 speak: none;
 font-size: 40px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-projector:before { content: "\e700"; }
.sf-icon-powerpoint:before { content: "\e701"; }
.sf-icon-onedrive:before { content: "\e702"; }
.default-stepper-section {
 width: 75%;
 margin: 0px auto;
 min-width: 85px;
 padding: 25px 0;
}
</style>

```

## TEMPLATE.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "PowerPoint", IconCss = "sf-icon-
powerpoint" });
 iconLabel.Add(new Step { Label = "Presentation", IconCss = "sf-icon-
projector" });
 iconLabel.Add(new Step { Label = "Backup", IconCss = "sf-icon-
onedrive" });

 ViewBag.IconLabel = iconLabel;

 return View();
}

```



## Tooltip in ASP.NET MVC Stepper control

The Stepper control supports tooltip to show additional information in the steps by setting the [ShowTooltip](#) property to `true`.

The tooltip appears when the user hovers over the step, providing the information such as the label or text. By default, the `ShowTooltip` property is `false`.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).ShowTooltip(true).Render()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-8;base64,AAEAAAAKAIAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAAA
YBg9jYQhUBIAAAAH4AAAAADmlheHABFgErAAABCAAAACBuYW1luF5THQAACTgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9AOaAAAAACAACAAAAAABAAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAQAAEAAAAAAAAAagAAAAAAMAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wbnCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAMAAEABAACAAMABQAAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKWEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFBwYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAGIBgcFBQQEAgH+CwEBAGQEBQUHBggICAK
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkJCACHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAGIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQCAQ0NDQwLCgoJCAGBQUDAgIDBQUGCAGJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgch/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQHCPCBAgQGBgkCgMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEAPDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfYQUEXEQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfYVFB1SEA8NCwoIBQQBAQQFCAoLDQ8QEH1UFRYXAAAAAQAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKWEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBgcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgchHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgchCAcICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAgICQgHCAcHBgYFBQQDAwIBAQIDAQwQFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWydAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgghBQE
BAIkCAGIEAwQFDA0SBwcGAgIBAQICBgCHBxYKQkJCACHBgUFBAQCAQEBAQIDAQwQFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdhIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHwYFhYVFRUVFAQCAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYGqgYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaQCAQEBAwKRNRi
HBqADChI1DQoFAgEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAuUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAQEAWQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAuUFBgYICakJCwsMKSckIiAeGxoYFhQ
```

```

TERAPDQwLCgkIDxsJBQUFBQQnEAKKCwWNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwcYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIIccsqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTyCmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAEAAAABAAAAAABAACAAQABAAAAAAACAACACAABAAAAAADAACADWABAAAAAAEAACAFgA
BAAAAAAFAAASAHQABAAAAAAGAACAkAABAAAAAAAKACwAlwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lubiBNZXRybyBtdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQBLAGWAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuaGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuaGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAa
AAAAAAYBAGEDAQQBBQEGAQCADXRYW5zcG9ydC12YW4LdXNlci1tb2RpZnkrRc2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hly2sAAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
}
</style>

```

## TOOLTIP.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address", IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment" });
 iconLabel.Add(new Step { Label = "Confirmation", IconCss = "sf-icon-success" });
}

```



```

 ViewBag.IconLabel = iconLabel;

 return View();
 }

```



### Tooltip template

You can use the [TooltipTemplate](#) property to specify a custom template for the tooltips, providing detailed information about the steps.

When hovering over the step, the current step model is passed in the template context, allowing you to include dynamic information about the step.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.TextLabelIcon).ShowTooltip(true)
.TooltipTemplate("#tooltip-template").Render()
<script id="tooltip-template" type="text/x-jsrender">

 ${value.text}

</script>
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSgcAAAEoAAAAVmNtYXCDelPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAAA
YbG9jYQhUblAAAAH4AAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAgAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOCc5wbnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAiwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxgDrAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz80OwEfDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAgQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQcHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAyGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQcHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQCAQ0NDQwLCgoJCAgGBQUdAgIDBQUgCgAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgch/iwNDawLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIHlY8PNS8CKwIPHQEHlwEjDwE1LwMjNz0BJZcfBTcfAg8BLwY
3OwEfAQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEa5FrBAQDBAMBawMDBgIDagIEBgY

```

```

GBxwCAwIBFQYGBAgFBgIWAqQHCPcBAgQGBggKCgsMDQ4PDxAQEBApDw4NDAsLCQgGBgQCAQEcbay
GCAkLCwWnDg8PEBAQEa8PDg0MCwoKCAYGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUEXeQDw0LCggFBAEXBhcFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAqIGBQMCAQQDBgZ
qBgoHBQMMDAMHBwMMAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQEcbayGCAkLCwWnDg8PEBA
QEa8PDg0MCwoKCAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBISEA8NCwoIBQQBAQQFCAoLDQ8QEHlUFRYXAAAAAQAaaaaa/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECAwQEBAYFBwYHCAgICAgICAcIBgcFBgQ
EBAICaSh6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICAkIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCAcICQgICAcHBwYGBQUEBAQICBAQICBAQY
GBwCHCAgICQgHCAcHBgYFBQQDAwIBAqIDAQFbQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCggHBQE
BAikCAgIEAwQFDA0SBwcGAgIBAqICBgCHBxYKQKjCAcHBgUFBAQCAgEBAQIDAQFbQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBwYCAQEBAQEBAgYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYGggYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBAqQCAQEBAwKRNRI
HBqADChI1DQoFagEBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBAiIDAUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAgICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBaQEAWQEAQEBaQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQIAGEBaf6RDAsLCQkICAYGBQUDAwIBAqIDAUFbGYICakJCwsMKsckIiAeGxoYfHq
TERAPDQwLCgkIDxsJBQUFbQqNEakKCwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKcGUPFz0
REakIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAEAAAAABAAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAAcADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAaCAKAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEECQAAAAI
AbQADAAEECQABAA4AbwADAAEECQACAA4AfQADAAEECQADAA4AiWADAAEECQAEAA4AmQADAAEECQA
FABYApwADAAEECQAGAA4AvQADAAEECQAKAFgAywADAAEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lvbiBNZXRybyBTdHVKaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABlAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIABnAGUAbgBlAHIAAYQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBuaGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBuaGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAaAaAaAaAaAaAaAaAaAaAaAaAaA
AAAAAAYBAgEDAQQBBQEGAQcADXRyYw5zcG9ydC12YW4LdXNlciltb2RpZnkcRc2hvcHBpbnctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');

```

```
font-weight: normal;
```

```
font-style: normal;
```

```
}
```

```
[class^="sf-icon-"], [class*=" sf-icon-"] {
```

```
font-family: 'Default' !important;
```

```
speak: none;
```

```
font-style: normal;
```

```
font-weight: normal;
```

```
font-variant: normal;
```

```
text-transform: none;
```

```
line-height: inherit;
```

```
-webkit-font-smoothing: antialiased;
```

```
-moz-osx-font-smoothing: grayscale;
```

```
}
```

```
.sf-icon-cart:before {
```

```
content: "\e710";
```

```
}
```

```
.sf-icon-payment:before {
```

```

 content: "\e706";
 }
 .sf-icon-success:before {
 content: "\e715";
 }
 #paymentStatus {
 padding: 10px;
 background-color: #e74d4d;
 color: white;
 border-radius: 5px;
 margin: 10px auto;
 text-align: center;
 font-weight: bold;
 max-width: 350px;
 }
 .stepper-section {
 width: 75%;
 margin: 0px auto;
 min-width: 85px;
 padding: 25px 0;
 }
}
</style>

```

### TOOLTIPTEMPLATE.CS

```

public ActionResult Demo()
{
 List<Step> textLabelIcon = new List<Step>();
 textLabelIcon.Add(new Step { Text = "Review your cart items", Label =
"Cart", IconCss = "sf-icon-cart" });
 textLabelIcon.Add(new Step { Text = "Enter your delivery address", Label =
"Delivery Address", IconCss = "sf-icon-transport" });
 textLabelIcon.Add(new Step { Text = "Complete your purchase securely",
Label = "Payment", IconCss = "sf-icon-payment" });
 textLabelIcon.Add(new Step { Text = "Verify your order details", Label =
"Confirmation", IconCss = "sf-icon-success" });

 ViewBag.TextLabelIcon = textLabelIcon;

 return View();
}

```



### Animation in ASP.NET MVC Stepper control

The Stepper progress state can be animated, smoothly transitioning from one step to another. You can customize the animation's [Duration](#) and [Delay](#), by using the [Animation](#) property.

You can disable the animation by setting the [Enable](#) property to `false`. By default, the value is `true`.

| Fields | Type | Description |

|-----|-----|-----|

| [Duration](#) | **number** | Specifies the duration of the animated transition for each step. The default value is **2000** milliseconds. |

| [Delay](#) | **number** | Specifies the delay to initiate the animated transition for each step in milliseconds. The default value is **0**. |

The example demonstrates the animation **Duration** and **Delay** settings for the Stepper.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.default).Animation(ViewBag.animation).Render()
```

### ANIMATION.CS

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 var animationSettings = new StepperAnimationSettings { Enable = true,
 Duration = 2000, Delay = 500 };
 ViewBag.default = defaultStepper;
 ViewBag.animation = animationSettings;
 return View();
}
```

## Globalization in ASP.NET MVC Stepper control

### Localization

The Localization library allows you to localize the default text content of the Stepper. You can change the static text content used for the **Optional** property to other cultures (Arabic, Deutsch, French, etc.) by defining the **Locale** value and its translation object.

| Locale key | en-US (default) |

|-----|-----|

| optional | Optional |

In this example, the **French** culture is set to Stepper and the default text is updated with the content defined by the locale key.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconLabel).Locale("fr-BE").Render()
<script>
 ej.base.L10n.load({
 'fr-BE': {
```

```
'stepper': {
 'optional': "facultatif"
}
});
</script>
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAIAIAAAAwAgTlMvMjlvSgcAAAEoAAAAVmNtYXCDeIPaAAABmAAAAF5nbHlmeWw
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAAYAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYw1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAABAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGly6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAABAAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAQAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAAAAagAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABOCc5wnCOC
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAAwADAAMAawADAAAAEABAACAAMABQAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxdAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWefDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwCHBQUEAwMBAQEBAwMEBQUHBwCICQgKCQoKCgkKJCAgIBGcFBQQEAgH+CwEBAGQEBQUHBGgICAK
JCgoKCQoICQgHBwCfBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ40DQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFawKE/eMBAQoJCQkKJCAChBgYFBAM
DAQEBAQMDBAUGBgCHCAkKJCQkKCgoJCQgICAgGBGQFAwICAgIDBAUFBgCHCAkKJCQoLCgkKJCQkIBwC
GBgUEAwMBAQEBAwMEBQYGBwCICQkKJCQoKCgkKJCAgIBwYGBAUDAgICAgMEBQUGBwCICQkKJCgGuAQI
GehYJBKYp/10ICAgGBQQAQ0NDQwLCgoJCAGGBQUDAgIDBQUGCAgJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBGcH/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCAoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgEaAgwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg40Dg0NDQwNCwwKCwoKCQgJBwCHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAgFBgIWAQGHCPcBAGQGBGgKCgsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUEXEQDw0LCgGfBAEXBhCFBAMDrxYWDQEBawUHCAsmDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECBAYGCAkLCwwNDg8PEBA
QEA8PDg0MCwoKCAyGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfYVFBISEA8NCwoIBQQAQQAQCAoLDQ8QEHUIFRYXAAAAAQAAAAA/QDRwA/AH8
AhwCRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwCTIRMnMSMhMxMhNSEDBzUha0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBAQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHBGcFBgQEBAMCAQECawQEBAYFBwYHCAgICAgICAgIBGcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwCGBGUFBAQCAgEBAQECAGQEBQUGBGcHBwg
ICAKIBwgHBwYGBQUEAwMCAQECawMEBQUGBGcCHCAICQgICACHwYGBQUEBAICAQEBQICBAQFBQY
GBwCHCAgICQgHCAChBgYFBQQDAwIBAQIDAQwFBQYGBwCIBwGB+wH+vQFABP5dOgGkAQEAAAAADAA
AAANKa5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDwYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAGoCAGENDAwKCgGHBQE
BAikCAGIEAwQFDA0SBwCgAgIBAQICBgCHBxYKQkKJCAChBgUFBAQCAQEBQIDAQwFBQYGBwCPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdYHwYCAQEBAGYHBwCWCgkKCAgHBwYFBQQAQAgE
BAQECawMEBAYfBgCHEBABAQED/qwUFRUVFRYWFhYWFxYXfHcXfxcWFxYXfHYWFhYVFRUVFQAQI
CBAUGCAGJCgsLDA0MDQ0NDBk2EQYgqgYGCEsZDQ0NDA0MCwsKCQgIBGUEAgIBaQQAQEBawRNRI
HBqADChI1DQoFAGeBAgMEBAoMEw8eTw4IVxkXCwkJBwYCOAIBaIIDAuUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECawMEBAUGBgYHCAgICQgHBwCGBGyYFBQQEBAyDIgICAQECaiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAGeAwQEAQICBAMFBQUGBwCICAgJBwGHBGc
GBGUFBAQEBQQAiAgEBAf6RDAsLCQkICAYGBQUDAwIBAQIDAuFbgYICAKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQQAeAKCwwNDxARExQWGBobHiAiJCcCoAMDawQECAXPRcKCGUPFz0
REAKIBAMDawMCAQICAwcYAwEaBwQBAgIAAAEAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
```

```

EJwwGAgIwXmMXFBIIcCsQKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYnmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAEAAAABAAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAACAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQ
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lubiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmcgU3luY2Z
lc2lubiBNZXRYbyBTdHVkaW93d3cuc3luY2Zlc2lubi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQB1AGwAdABEAGUAZgBhAHUAbAB0AFYAZQByAHMAaQBvAG4AIAAxAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAbgBlAHIAZQB0AGUAZAAGAHUAcwBpAG4AZwA
gAFMAeQBvAGMAZgBlAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBvAGMAZgBlAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAoooooAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAYBAgEDAQQBBQEGAQcADXRYW5zcG9ydC12YW4LdXNlci1tb2RpZnkrC2hvcHBpbmctY2F
ydfF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('true');

 font-weight: normal;
 font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
 font-family: 'Default' !important;
 speak: none;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: inherit;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
 content: "\e710";
}
.sf-icon-transport:before {
 content: "\e702";
}
.sf-icon-payment:before {
 content: "\e706";
}
.sf-icon-success:before {
 content: "\e715";
}
}
.stepper-section {
 margin: 50px 100px;
}
}
</style>

```

## LOCALIZATION.CS

```

public ActionResult Demo()
{
 List<Step> iconLabel = new List<Step>();
 iconLabel.Add(new Step { Label = "Cart", IconCss = "sf-icon-cart" });
 iconLabel.Add(new Step { Label = "Delivery Address", IconCss = "sf-icon-transport" });
 iconLabel.Add(new Step { Label = "Payment", IconCss = "sf-icon-payment", Optional = true });
}

```

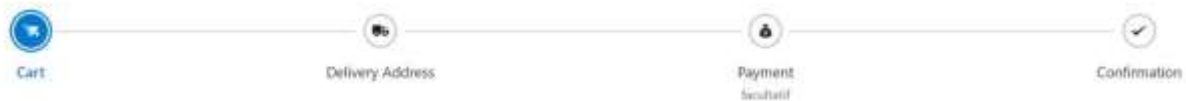
```

 iconLabel.Add(new Step { Label = "Confirmation", IconCss = "sf-icon-
 success" });

 ViewBag.IconLabel = iconLabel;

 return View();
 }

```



### RTL

RTL provides an option to switch the text direction and layout of the Stepper control from right to left by setting the [EnableRtl](#) property to **true**.

### CSHTML

```

@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.IconText).EnableRtl(true).Rende
r()
<style>
 @@font-face {
 font-family: 'Default';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj1vSgcAAAEoAAAAVmNtYXCDelIPaAAABmAAAAF5nbHlmEwr
+pwAAAggAAAJQaGVhZCYp2+EAAADQAAANmhoZWEIUQQHAAAArAAAACRobXR4GAAAAAAYAAAAA
YbG9jYQhUBlAAAAH4AAAAADm1heHABFgErAAABCAAAACBuYW1luF5THQAACtgAAAIlcG9zdJ8LuoM
AAA0AAAAAbwABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAABgABAAAAQAARxT6wV8
PPPUACwQAAAAAOGLy6UAAAAA4YvLpQAAAAAD9A0aAAAAACAACAAAAAAAAAAAAEAAAAGAR8ABgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wLnFQQAAAAAXAQAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAAAAGAAAAAUAAMAAQAAABQABABKAAAADAAIAAIABoCc5wnCoc
Q5xX//wAA5wLnBucI5xDnFf//AAAAAAAAAAAAAAAAABAawADAAMAawADAAAAEABAACAAMABQAAAAA
AAAEQAIwC3AQkBGgAAAAFAAAAAAP0A18APwB/AIkAxdRAAABHw8/Dy8OKwEPDQUfDz8PLw4rAQ8
NAR8FFSM1JxEfBz8OOWefDjM/BzUnIw8GATM/Dx8PMxEhAq8BAQIEBAUFbWYICAgJCQoKCgkKCAk
IBwcHBQUEAwMBAQEBAwMEBQUHBwcICQgKCQoKCgkJCAgIBgcFBQQEAgH+CwEBAGQEBQUHBggICAk
JCgoKCQoICQgHBwcFBQQDAwEBAQEDAwQFBQCHBwgJCAoJCgoKCQkICAgGBwUFBAQCAQJ8AwUIWAw
D3n0BAwMGBgYICAMEBQYHBwkJCgsLDA0NDQ4ODQ4MDAwLCgkJCAYGBQMDKAgIBwYFBAECvLsICAY
HBQMD/beAAwQFBQCHCAkKCgsLDA0MDg0NDQwLCwsJCQkHBwUFAwKE/eMBAQoJCQkJCAcHBgYFBAM
DAQEBAQMDBAUGBgCHCAkJCQkKCgoJCQgICAcGBgQFAwICAgIDBAUFBgCHCAkJCQoLCgkJCQkIBwc
GBgUEAwMBAQEBAwMEBQYGBwcICQkJCQoKCgkJCAgIBwYGBAUDAgICAgMEBQUGBwcICQkJCgGuAQI
GehYJBKYp/10ICAcGBQQAQ0NDQwLCgoJCAgGBQUDAgIDBQUGCagJCgoLDA0NDQECBAUGBwQI1fo
BAGQFBgch/iwNDAwLCwoJCQgHBgUEAwEBAQEDBAUGBwgJCQoLCwwMDQJJAAAABgAAAAAD8wOWAAY
AQgBaAGwArQDuAAABZcfAwUhLwIH1y8PNS8CKwIPHQEHLwEjDwE1LwMjNz0BJzcfBTcfAg8BLwY
3OwEfaQcVHw8/Dy8PDw4BFR8PPw8vDw8OAxEWBgEDAgb8/wNuBAUEDQsVFBQTEhEPDw0GCwoIBgQ
CFhITE+wQDw8PDg4ODg0NDQwNCwwKCwoKCQgJBwcHBgYEBQMEA5FrBAQDBAMBawMDBgIDagIEBgY
GBxwCAwIBFQYGBAGFBgIWAQgHCPcBAGQGBgKCGsMDQ4PDxAQEBAPDw4NDAsLCQgGBgQCAQECCBAY
GCAkLCwwNDg8PEBAQEA8PDg0MCwoKCAyGBAL+KgEEBQgKCw0PEBETFBQWFxcXfHYUFBMREQ4NDak
IBgMBAQMGCakMDQ4RERMUFBYWFxcXfHQUExEQDw0LCggFBAEXBhcfBAMDrxYWDQEBawUHCAmsMDQ4
IERESFBQUFQQDAgECAGMEBAUGBgYIBwgJCQoKCwsLDAwMDQ0ODQ4PDgEZawIBAQIGBQMCAQQDBgZ
qBgOHBQMDMAMHBwMWAQICBQYKChYCB1wICBAPDw4NDAsLCQgGBgQCAQECCBAYGCAkLCwwNDg8PEBA

```

```

QEA8PDg0MCwoKcAYGAwMBAQMDBgYICgoLDA0ODw8QATMLDBYVFRQSERAPDQsKCAUEAQEEBQgKCw0
PEBESFBUVFhcXfXyVFBiSEA8NCwoIBQQBAQQFCAoLDQ8QEHuIFRYXAAAAAAQAAAAAA/QDRwA/AH8
AhwcRAAABFR8OPw49AS8NKwEPDQUVHw4/Dj0BLw0rAQ8NEwcTIRMnMSMhMxMhNSEDBzUhA0YBAgM
EBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgEBAgMEBAQGBQcGBwgICAgICAgIBwYHBQYEBaQDAgH
+aAICAgQEBAYFBwYIBwgICAgICAgHbGcFBgQEBAMCAQECAwQEBAYFBwYHCAGICAgICAcIBgcFBgQ
EBAICAsH6jAFKjPpu/Z3NwgJZ/dzDAf8AAQkICAgHBwcGBgUFBAQCAgEBAQECAgQEBQUGBgCHBwg
ICaKIBwgHBwYGBQUEAwMCAQECAwMEBQUGBgCHCaICQgICAcHBwYGBQUEBAICAQEBAQICBAQFBQY
GBwcHCAGICQgHCACHBgYFBQQDAwIBAQIDAQWFBQYGBwcIBwgB+wH+vQFABP5dOgGkAQEAAAADAAA
AAANkA5oAnQDxAR4AAAEzHwEdAR8HFQ8DIy8HDWYdAR8WDw0dAQ8BKwIvAT0BLwc9AT8COWEfBj8
HLxc/DTU/AwEfDjsBPxEvFiMPFR8BEw8CFR8HMz8HNS8GIw8ELwQrAQ8BAgoCAgENDAwKCgGHBQE
BAiKCAgIEAwQFDA0SBwcGAgIBAQICBgCHBxYKcQkJCAcHBgUFBAMCAQEBAQIDAQWFBQYGBwcPEQE
CAhUCAQINDAsLCQgHBQICKQICAgQDBAULdHIBWYCAQEBAQEBAGYHBwcWCgkKCAgHBwYFBQQDAgE
BAQECAwMEBAYFBgcHEBABAQED/qwUFRUVFYWFhYWFxYXfHcXfxcWFxYXfHcYWFhYVFRUVFAQCAQI
CBAUGCAgJCgsLDA0MDQ0NDBk2EQYgGqYGCEsZDQ0NDA0MCwsKCQgIBgUEAgIBaqQCAQEBawkrNRRI
HBqADChI1DQoFAGEBAGMEBAoMEw8eTw4IVxkXCwkJBWYCOAIBaiIDAUGBwgJCgICAQENAQEFawI
DAgECAGMFAwMEBAUDBAMFAwIBAQECAwMEBAUGBgYHCAGICQgHBwcGBgYFBQQEBAYDIgICAQECAiI
CBAUGBwgJCQMBAGEMAQUDAwIDAQICBAQDBAQEBAGAEAwQEAgEBAQICBAMFBQUGBwcICAgJBwgHBgc
GBgUFBAQEBAQqIAGEBaf6RDAsLCQkICAYGBQUDAwIBAQIDAQWFBgYICAKJCwsMKSckIiAeGxoYFhQ
TERAPDQwLCgkIDxsJBQUBQqNEAKKCwNDxARExQWGBobHiAiJCcCoAMDawQECA8XPRcKCgUPFz0
REAKIBAMDawMCAQICAwCYAwEaBwQBAGIAAAEAAAAAA/MDNAA0AAABDwQvAw8EHwQ/ETUnIw8LAYs
EJwwGAgIwXmMXFBIICCSqKaEqRUclJSYnJykpKiosLC4GFgsCAWMhISIiIiIjIkJAPRwB8AQmCQM
BARQuNgsMDgcIJCYNmyZOTycmJiYlJSQjIiIgHwULCAMCAQ4RERITFBUVKy0tFgAAABIA3gABAAA
AAAAAAAAAAAAAABAAAAAABAAcAAQABAAAAAAACAACACAABAAAAAADAACADwABAAAAAAAEAAcAFgA
BAAAAAAFAAAsAHQABAAAAAAGAACAkAABAAAAAAAKACwALwABAAAAAALABIAWwADAAEEECQAAAAI
AbQADAAEEECQABAA4AbwADAAEEECQACAA4AfQADAAEEECQADAA4AiWADAAEEECQAEAA4AmQADAAEEECQA
FABYApwADAAEEECQAGAA4AvQADAAEEECQAKAFgAywADAAEEECQALACQBIyBEZWZhdWx0UmVndWxhckR
lZmFlbHREZWZhdWx0VmVyc2lvbiAxLjBEZWZhdWx0Rm9udCBnZW5lcmF0ZWQgdXNpbmctU3luY2Z
lc2lvbiBNZXRybyBTdHVkaW93d3cuc3luY2Zlc2lvbi5jb20AIABEAGUAZgBhAHUAbAB0AFIAZQB
nAHUAbABhAHIArABLAGYAYQBLAGwAdABEAGUAZgBhAHUAbAB0AFYAZQBByAHMAaQBvAG4AIAAxAAC4
AMABEAGUAZgBhAHUAbAB0AEYAbwBuAHQAIAABnAGUAAbgBIAHIAAYQB0AGUAZAAGAHUAACwBpAG4AZwA
gAFMAeQBBAAGMAZgBIAHMAaQBvAG4AIAABNAGUAdABYAG8AIAABTAHQAdQBkAGkAbwB3AHcAdwAuAHM
AeQBBAAGMAZgBIAHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAaAAAAAaAAAAAaAAAAAaAAAAAaAAAAA
AAAAAAYBAGEDAQBBQEGAQCADXRYW5zcG9ydC12YW4LdXNlcil1tb2RpZnZkR2hvcHBpbmctY2F
ydF8wMS0Lc3BlbmQtbW9uZXkFY2hlY2sAAAA=) format('trueType');

font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'Default' !important;
speak: none;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: inherit;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
}
.sf-icon-cart:before {
content: "\e710";
}
.sf-icon-transport:before {
content: "\e702";
}
.sf-icon-payment:before {
content: "\e706";
}
}

```



```

 .sf-icon-success:before {
 content: "\e715";
 }
 .stepper-section {
 margin: 50px 100px;
 }
 }
</style>

```

## RTL.CS

```

public ActionResult Demo()
{
 List<Step> iconText = new List<Step>();
 iconText.Add(new Step { Text = "Cart", IconCss = "sf-icon-cart" });
 iconText.Add(new Step { Text = "Delivery Address", IconCss = "sf-icon-transport" });
 iconText.Add(new Step { Text = "Payment", IconCss = "sf-icon-payment" });
 iconText.Add(new Step { Text = "Confirmation", IconCss = "sf-icon-success" });

 ViewBag.IconText = iconText;

 return View();
}

```



## Accessibility in ASP.NET MVC Stepper control

Accessibility is achieved in the Stepper control through WAI-ARIA standard and keyboard navigations. The Stepper control can be effectively accessed through assistive technologies such as screen readers.

### Keyboard interaction

The Stepper control is interactive with the below keyboard shortcuts.

#### | Keyboard shortcuts | Actions |

| --- | --- |

| Up Arrow | Focuses the previous step in a vertical Stepper. |

| Down Arrow | Focuses the next step in a vertical Stepper. |

| Left Arrow | Focuses the previous step in a horizontal Stepper. |

| Right Arrow | Focuses the next step in a horizontal Stepper. |

| Home | Focuses on the first step of the Stepper. |

| End | Focuses on the last step of the Stepper. |

| Enter / Space | Activates the currently focused step. |

### ARIA attribute

The following ARIA attributes are used in Stepper control based on its state.

| Properties           | Functionality                                                                               |
|----------------------|---------------------------------------------------------------------------------------------|
| -----                | -----                                                                                       |
| <b>aria-label</b>    | Attribute provide a descriptive text that labels the interactive element for accessibility. |
| <b>aria-current</b>  | Attribute denotes the currently active step in the Stepper.                                 |
| <b>aria-disabled</b> | Indicates when the step is disabled and cannot be interacted.                               |

### Events

This section describes the Stepper events that will be triggered when an appropriate actions are performed. The following events are available in the Stepper control.

#### Created

The Stepper control triggers the [Created](#) event when the control rendering is completed.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.DefaultStepper).Created("function(args) { createdEvent() }").Render()
<script>
 function createdEvent() {
 // your required action here..
 }
</script>
```

#### CREATED.CS

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 ViewBag.DefaultStepper = defaultStepper;
 return View();
}
```

#### StepChanged

The Stepper control triggers the [StepChanged](#) event after the active step is changed.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.DefaultStepper).StepChanged("function(args) { stepChangedEvent(args) }").Render()
<script>
 function stepChangedEvent(args) {
 alert("Step Changed from "+args.previousStep + " to " +
args.activeStep)
```

```
}
</script>
```

### STEPCHANGED.CS

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 ViewBag.DefaultStepper = defaultStepper;
 return View();
}
```

### StepChanging

The Stepper control triggers the [StepChanging](#) event before the active step change.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.DefaultStepper).StepChanging("function(args) { stepChangingEvent(args) }").Render()
<script>
 function stepChangingEvent(args) {
 alert("Step Changing from "+args.previousStep + " to " +
args.activeStep)
 }
</script>
```

### STEPCHANGING.CS

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 ViewBag.DefaultStepper = defaultStepper;
 return View();
}
```

### StepClick

The Stepper control triggers the [StepClick](#) event when the step is clicked.

### CSHTML

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.DefaultStepper).StepClick("function(args) { stepClickEvent(args) }").Render()
<script>
```

```
function stepClickEvent(args) {
 // your required action here..
}
</script>
```

**STEPCLICK.CS**

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 ViewBag.DefaultStepper = defaultStepper;
 return View();
}
```

**BeforeStepRender**

The Stepper control triggers the [BeforeStepRender](#) event before rendering each step.

**CSHTML**

```
@using Syncfusion.EJ2.Navigations
@Html.EJS().Stepper("stepper").Steps(ViewBag.DefaultStepper).BeforeStepRender(
 "function(args) { beforeStepRenderEvent(args) }").Render()
<script>
 function beforeStepRenderEvent(args) {
 // your required action here..
 }
</script>
```

**BEFORESTEPRENDER.CS**

```
public ActionResult Demo()
{
 List<Step> defaultStepper = new List<Step>();
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 defaultStepper.Add(new Step { });
 ViewBag.DefaultStepper = defaultStepper;
 return View();
}
```

**Stock Chart****Getting Started with ASP.NET MVC Stock Chart Control**

This section briefly explains about how to include [ASP.NET MVC Stock Chart](#) control in your ASP.NET MVC application using Visual Studio.

## Prerequisites

### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Stock Chart control

Now, add the Syncfusion ASP.NET MVC Stock Chart control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@(Html.EJS().StockChart("container").Load("load").Series(sr => { sr.Add();
}).Render())
```

### Populate Stock Chart With Data

This section explains how to plot below JSON data to the Stock Chart.

Add a series object to the chart by using [Series](#) property and then set the JSON data to [DataSource](#) property.

Since the JSON contains category data, set the [valueType](#) for horizontal axis to Category. By default, the axis valueType is Numeric.

#### CSHTML

```
@(Html.EJS().StockChart("container").Load("stockload")
 .Series(sr =>
 {
 sr.Name("Apple").Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).
 XName("x").High("high").Low("low").Open("open").Close("close").Add();
 })
 .Render())
<script src="~/Scripts/chart/financial-data.js"></script>
<script>
 var data = chartData;
 function stockload(args) {
 args.stockChart.series[0].dataSource = data;
 }
</script>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Stock Chart control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

**Note:** You can refer to our [ASP.NET MVC Stock Chart](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Stock Chart example](#) that shows you how to present and manipulate data.

<!-- markdownlint-disable MD036 -->

## Working with Data

Chart can visualise data bound from local or remote data.

### Local Data

You can bind a simple JSON data to the chart using [DataSource](#) property in series.

### CSHTML

```
@using Syncfusion.EJ2;
@Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add();
 })
 .ChartArea(area => area.Border(br=>br.Width(0)))
```

```

.Border(br => br.Width(1))
.Render()
<script>
var data = [
 { date: new Date('2012-04-02'), open: 85.9757, high: 90.6657, low:
85.7685, close: 90.5257, volume: 660187068 },
 { date: new Date('2012-04-09'), open: 89.4471, high: 92, low:
86.2157, close: 86.4614, volume: 912634864 },
 { date: new Date('2012-04-16'), open: 87.1514, high: 88.6071, low:
81.4885, close: 81.8543, volume: 1221746066 },
 { date: new Date('2012-04-23'), open: 81.5157, high: 88.2857, low:
79.2857, close: 86.1428, volume: 965935749 },
 { date: new Date('2012-04-30'), open: 85.4, high: 85.4857, low:
80.7385, close: 80.75, volume: 615249365 },
 { date: new Date('2012-05-07'), open: 80.2143, high: 82.2685, low:
79.8185, close: 80.9585, volume: 541742692 },
 { date: new Date('2012-05-14'), open: 80.3671, high: 81.0728, low:
74.5971, close: 75.7685, volume: 708126233 },
 { date: new Date('2012-05-21'), open: 76.3571, high: 82.3571, low:
76.2928, close: 80.3271, volume: 682076215 },
 { date: new Date('2012-05-28'), open: 81.5571, high: 83.0714, low:
80.0743, close: 80.1414, volume: 480059584 },
 { date: new Date('2012-06-04'), open: 80.2143, high: 82.9405, low:
78.3571, close: 82.9028, volume: 517577005 },
]
</script>

```

### LOCAL-DATA.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

See Also

- [Series Types](#)

### Stock Chart Dimensions

Size for Container

Chart can render to its container size. You can set the size via inline or CSS as demonstrated below.

### CSHTML



```

<script src="~/financial-data.js"></script>
<div class="control-section">
 <div id="control-container" >
 @(Html.EJS().StockChart("container").Title("AAPL Stock
Price").Width("650").Height("350")
 .Series(sr =>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render())
 <script>
 var data = window.chartData;
 </script>
}

```

### SIZE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### Size for Stock Chart

You can also set size for chart directly through [Width](#) and [Height](#) properties.

<!-- markdownlint-disable MD036 -->

### In Pixel

<!-- markdownlint-disable MD036 -->

You can set the size of chart in pixel as demonstrated below.

### CSHTML

```

<script src="~/financial-data.js"></script>
 @(Html.EJS().StockChart("container").Title("AAPL Stock
Price").Width("650px").Height("350px")
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();

```

```

 })
 .Render ()

</script>
<script>
 var data = window.chartData;
</script>

```

**PIXEL.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

**In Percentage**

By setting value in percentage, chart gets its dimension with respect to its container. For example, when the height is '50%', chart renders to half of the container height.

**CSHTML**

```

<script src="~/financial-data.js"></script>
 @(Html.EJS().StockChart("container").Title("AAPL Stock
Price").Width("80%").Height("90%")
 .Series(sr
=>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
 ();
 })
 .Render())

<script>
 var data = window.chartData;
</script>

```

**PERCENTAGE.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{

```

```

public partial class StockChartController : Controller
{
 public IActionResult Default()
 {
 return View();
 }
}

```

**Note:** When you do not specify the size, it takes 450px as the height and window size as its width.

## Axis types

### DateTime axis

DateTime axis uses date time scale and displays the date time values as axis labels in the specified format. To use DateTime axis, set the [ValueType](#) of axis to `DateTime`.

### CSHTML

```

<script src="~/financial-data.js"></script>
@(Html.EJS().StockChart("container").Title("AAPL Stock
Price").IndicatorType(new List<Object>() { }).ExportType(new List<Object>()
{ }).TrendlineType(new List<Object>() { })
 .PrimaryXAxis(xaxis
=>xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render())

<script>
 var data = window.chartData;
</script>

```

### DATETIME.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### DateTimeCategory axis

DateTimeCategory axis in the stock chart is used to display only business days. To use DateTimeCategory axis, set the [ValueType](#) of axis to [DateTimeCategory](#).

### CSHTML

```
@(Html.EJS().StockChart("container").Load("stockload")
 .PrimaryXAxis(xaxis
=>xaxis.MajorGridLines(mg=>mg.Width(0)).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTimeCategory)
 .CrosshairTooltip(ct=>ct.Enable(true))
).Series(sr =>
 {

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("y").Add();
 })
 .Crosshair(cr=>cr.Enable(true))
 .Tooltip(tp => tp.Enable(true).Header("AAPL Stock Price")).Title("AAPL Stock Price")
 .Render())
<script>
 let datetimeCategoryData = [
 {x: new Date(2021, 1, 11) }, {x: new Date(2021, 1, 12) }, {x: new Date(2021, 1, 13) }, {x: new Date(2021, 1, 14) }, {x: new Date(2021, 1, 15) },
 },
 {x: new Date(2021, 1, 19) }, {x: new Date(2021, 1, 20) }, {x: new Date(2021, 1, 21) }, {x: new Date(2021, 1, 22) }, {x: new Date(2021, 3, 1) },
 },
 {x: new Date(2021, 3, 2) }, {x: new Date(2021, 4, 1) }, {x: new Date(2021, 4, 5) }, {x: new Date(2021, 4, 6) }, {x: new Date(2021, 4, 7) },
 {x: new Date(2021, 4, 11) }, {x: new Date(2021, 4, 13) }, {x: new Date(2021, 4, 15) }, {x: new Date(2021, 4, 16) }, {x: new Date(2021, 4, 17) },
 },
 {x: new Date(2021, 4, 18) }, {x: new Date(2021, 4, 20) }, {x: new Date(2021, 4, 21) }, {x: new Date(2021, 4, 23) }, {x: new Date(2021, 4, 25) },
 },
 {x: new Date(2021, 5, 1) }, {x: new Date(2021, 5, 2) }, {x: new Date(2021, 5, 6) }, {x: new Date(2021, 5, 7) }, {x: new Date(2021, 5, 8) },
 {x: new Date(2021, 5, 11) }, {x: new Date(2021, 5, 15) }, {x: new Date(2021, 5, 18) }, {x: new Date(2021, 5, 20) }, {x: new Date(2021, 5, 25) },
 },
 {x: new Date(2021, 6, 1) }, {x: new Date(2021, 6, 2) }, {x: new Date(2021, 6, 3) }, {x: new Date(2021, 6, 4) }, {x: new Date(2021, 6, 5) },
 {x: new Date(2021, 6, 10) }, {x: new Date(2021, 6, 11) }, {x: new Date(2021, 6, 12) }, {x: new Date(2021, 6, 13) }, {x: new Date(2021, 6, 15) },
 },
 {x: new Date(2021, 6, 16) }, {x: new Date(2021, 6, 17) }, {x: new Date(2021, 6, 18) }, {x: new Date(2021, 6, 19) }, {x: new Date(2021, 6, 20) },
 }
]
 function stockload(args) {
 let series2 = [];
 let point2;
 for (var i = 0; i < 46; i++) {
 point2 = {
 x: datetimeCategoryData[i].x,
```

```

 y: getRandomInRange(120, 130),
 high: getRandomInRange(88, 92),
 low: getRandomInRange(76, 86),
 open: getRandomInRange(75, 85),
 close: getRandomInRange(85, 90),
 volume: getRandomInRange(660187068, 965935749)
 };
 series2.push(point2);
}
args.stockChart.series[0].dataSource = series2;
}
function getRandomInRange(min, max) {
 const randomDecimal = Math.random();
 const randomValue = randomDecimal * (max - min) + min;
 return randomValue;
}
</script>

```

### DATETIMECATEGORY.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult DateTimeCategory()
 {
 return View();
 }
 }
}

```

### Logarithmic axis

<!-- markdownlint-disable MD033 -->

Logarithmic axis uses logarithmic scale and it is very useful in visualizing data, when it has numerical values in both lower order of magnitude (eg:  $10^{-6}$ ) and higher order of magnitude (eg:  $10^6$ ). To use Logarithmic axis, set the [ValueType](#) of axis to [Logarithmic](#).

### CSHTML

```

<script src="~/financial-data.js"></script>
@ (Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .PrimaryXAxis(xaxis
=>xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .PrimaryYAxis(yaxis
=>yaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.Logarithmic))
 .Series(sr
=>
{

```

```

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").YName("volume").Add();
 })
 .Render())

<script>
 var data = window.chartData;
</script>

```

## LOG.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

See also

- [Axis Customization](#)

## Axis Customization

<!-- markdownlint-disable MD034 -->

### Axis Crossing

An axis can be positioned in the chart area using [CrossesAt](#) and [CrossesInAxis](#) properties. The [CrossesAt](#) property specifies the values (datetime, numeric, or logarithmic) at which the axis line has to be intersected with the vertical axis or vice-versa, and the [CrossesInAxis](#) property specifies the axis name with which the axis line has to be crossed.

## CHTML

```

@Html.EJS().StockChart("container").PrimaryXAxis(xaxis
=>xaxis.CrossesAt(160))
 .Series(sr =>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 <script src="~/financial-data.js"></script>
 .Render()
 <script>
 var data = window.chartData;

```

```
</script>
```

### AXIS-CROSS.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Title

You can add a title to the axis using [Title](#) property to provide quick information to the user about the data plotted in the axis. Title style can be customized using [TitleStyle](#) property of the axis.

### CSHTML

```
@(Html.EJS().StockChart("container").Title("AAPL Stock Price")
.Series(sr =>
{
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
());
<script src="~/financial-data.js"></script>
.Render())
<script>
var data = window.chartData;
</script>
```

### TITLE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

```
}
}
```

### Tick Lines Customization

You can customize the **Width**, **Color** and **Size** of the minor and major tick lines, using [MajorTickLines](#) and [MinorTickLines](#) properties in the axis.

#### CSHTML

```
@Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .PrimaryXAxis(xaxis
=>xaxis.MajorTickLines(mg=>mg.Color("blue").Width(5)
).MinorTickLines(mg=>mg.Color("red").Width(0)))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("green").Width(5)
).MinorTickLines(mg=>mg.Color("red").Width(0)))
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render()

<script src="~/financial-data.js"></script>
<script>
 var data = window.chartData;
</script>
```

#### TICK.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Grid Lines Customization

You can customize the **Width**, **Color** and **DashArray** of the minor and major grid lines, using [MajorGridLines](#) and [MinorGridLines](#) properties in the axis.

#### CSHTML

```
@Html.EJS().StockChart("container").Title("AAPL Stock Price")
```



```

 .PrimaryXAxis(xaxis
=>xaxis.MajorGridLines(mg=>mg.Color("blue").Width(1)
).MinorGridLines(mg=>mg.Color("red").Width(0)))
 .PrimaryYAxis(yaxis
=>yaxis.MajorGridLines(mt=>mt.Color("green").Width(1)
).MinorGridLines(mg=>mg.Color("red").Width(0)))
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render()

<script src="~/financial-data.js"></script>
<script>
 var data = window.chartData;
</script>

```

## GRID.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

## Multiple Axis

In addition to primary X and Y axis, we can add n number of axis to the chart. Series can be associated with this [Axis] (<https://help.syncfusion.com/cr/aspnetcore-js2/Syncfusion.EJ2.Charts.StockChartAxis.html>), by mapping with axis's unique name.

## CSHTML

```

<script src="~/goog.js"></script>
<script src="~/googl.js"></script>
@Html.EJS().StockChart("container").Title("Multiple Series")
 .PrimaryXAxis(xaxis
=>xaxis.MajorGridLines(mg=>mg.Width(0)).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).DataSource("data1").XName
("x").YName("close").Add();

```

```

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Column).YAxisName("yAxis").DataSource("data2").XName("x").YName("close").Add();
 }).Axes(ax =>
 {
 ax.OpposedPosition(true).RowIndex(0).Name("yAxis").Add();
 }
)

.Render()

<script>
 var data1 = window.goog;
 var data2 = window.googl;
</script>

```

### MULTIPLE.CS

```

public ActionResult Index()
{
 List<MultipleAxesChartData> chartData = new
List<MultipleAxesChartData>
 {
 new MultipleAxesChartData { x = "Sun", y = 35, y1 = 30 },
 new MultipleAxesChartData { x = "Mon", y = 40, y1 = 28 },
 new MultipleAxesChartData { x = "Tue", y = 80, y1 = 29 },
 new MultipleAxesChartData { x = "Wed", y = 70, y1 = 30 },
 new MultipleAxesChartData { x = "Thu", y = 65, y1 = 33 },
 new MultipleAxesChartData { x = "Fri", y = 55, y1 = 32 },
 new MultipleAxesChartData { x = "Sat", y = 50, y1 = 34 }
 };
 ViewBag.dataSource = chartData;
 return View();
}

public class MultipleAxesChartData
{
 public string x;
 public double y;
 public double y1;
}

```

### Inversed Axis

<!-- markdownlint-disable MD033 -->

When an axis is inversed, highest value of the axis comes closer to origin and vice versa. To place an axis in inversed manner set this property [IsInversed](#) to true.

### CSHTML

```

@Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .PrimaryXAxis(xaxis
=>xaxis.IsInversed(true))..PrimaryYAxis(yaxis =>yaxis.IsInversed(true))
 .Series(sr
=>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
 ();
 }
)

```

```

 })
 .Render ()

<script src="~/financial-data.js"></script>
<script>
 var data = window.chartData;
</script>

```

### INVERSED.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### Opposed Position

<!-- markdownlint-disable MD012 -->

To place an axis opposite from its original position, set [OpposedPosition](#) property of the axis to true.

<!-- markdownlint-disable MD012 -->

### CSHTML

```

@Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .PrimaryXAxis(xaxis =>xaxis.OpposedPosition(true))
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render ()

<script src="~/financial-data.js"></script>

<script>
 var data = window.chartData;
</script>

```

### OPPOSED.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

```

```
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

## Stock Chart Series Types

Essential JS 2 StockChart Stock Chart supports 6 major types of series namely **Line**, **Spline**, **Hilo**, **HiloOpenClose**, **Hollow Candle** and **Candle** . By using the series dropdown button you can navigate between the above listed series types.

<!-- markdownlint-disable MD036 -->

### Line

To render a line series, use series **Type** as **Line**.

### Candle

To render a candle series, use series **Type** as **Candle** .

### HollowCandle

To render a hollowcandle series, use series **Type** as **Candle** and set **EnableSolidCandle** as false.

### Spline

To render a spline series, use series **Type** as **Spline**.

### Hilo

To render a hilo series, use series **Type** as **Hilo** .

### HiloOpenClose

To render a hiloOpenClose series, use series **Type** as **HiloOpenClose**.

### CSHTML

```
<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").IndicatorType(new List<Object>() { }).ExportType(new List<Object>()
{ }).TrendlineType(new List<Object>() { })

=>

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();

.Render())
```

```
<script>
 var data = window.chartData;
</script>
```

## CANDLE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

## Trendlines

Trendlines are used to show the direction and speed of price.

Stock Chart supports 6 types of trendlines namely

Linear, Exponential, Logarithmic, Polynomial, Power, Moving Average. By using trendline dropdown button you can add or remove the required trendline type.

### Linear

A linear trendline is a best fit straight line that is used with simpler data sets. To render a linear trendline, use trendline [Type](#) as Linear.

### Exponential

An exponential trendline is a curved line that is most useful when data values rise or fall at increasingly higher rates. You cannot create an exponential trendline, if your data contains zero or negative values.

To render a exponential trendline, use trendline [Type](#) as Exponential.

### Logarithmic

A logarithmic trendline is a best-fit curved line that is most useful when the rate of change in the data increases or decreases quickly and then levels out. A logarithmic trendline can use negative and/or positive values.

To render a logarithmic trendline, use trendline [Type](#) as Logarithmic.

### Polynomial

A polynomial trendline is a curved line that is used when data fluctuates.

To render a polynomial trendline, use trendline [Type](#) as Polynomial.

**PolynomialOrder** used to define the polynomial value.

### Power

A power trendline is a curved line that is best used with data sets that compare measurements that increase at a specific rate.

To render a power trendline, use trendline [Type](#) as **Power**.

### Moving Average

A moving average trendline smoothen out fluctuations in data to show a pattern or trend more clearly.

To render a moving average trendline, use trendline [Type](#) as **MovingAverage**.

**Period** property defines the period to find the moving average.

### CSHTML

```
<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").IndicatorType(new List<Object>() { }).ExportType(new List<Object>()
{ }).SeriesType(new List<Object>() { })

=>

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();

<script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.series[0].trendlines = [{ type: 'MovingAverage',
enableTooltip: false }]
 }
</script>
```

### TRENDLINES.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Customization of Trendline

The [Fill](#) and [Width](#) properties are used to customize the appearance of the trendline.

#### CSHTML

```
<script src="~/financial-data.js"></script>
 @(Html.EJS().StockChart("container").Title("AAPL Stock
Price").IndicatorType(new List<Object>() { }).ExportType(new List<Object>()
{ }).SeriesType(new List<Object>() { })
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Trendlines(tr =>
tr.Type(Syncfusion.EJ2.Charts.TrendlineTypes.MovingAverage).EnableTooltip(false).Fill("red").Width(2)).Add();
 })

 .Crosshair(cr=>cr.Enable(true).Line(line=>line.Color("green").Width(2)))
 })
 .Render())

<script>
 var data = window.chartData;
</script>
}
```

#### CUSTOMTRENDLINES.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

<!-- markdownlint-disable MD036 -->

### Technical Indicators

A technical indicator is a mathematical calculation based on historic price, volume or open interest information that aims to forecast financial market direction.

Stock Chart supports 10 types of technical indicators namely Accumulation Distribution, ATR, EMA, SMA, TMA, Momentum, MACD, RSI, Stochastic, Bollinger Band. By using indicator dropdown box you can add and remove the required indicators types.

### Accumulation Distribution

Accumulation Distribution combines price and volume to show how money may be flowing into or out of stock.

To render a Accumulation Distribution Indicator, use indicator [Type](#) as `AccumulationDistribution`.

To calculate the signal line [Volume](#) field is additionally added with `DataSource`.

### Average True Range (ATR)

ATR measures the stock volatility by comparing the current value with the previous value.

To render a Average True Range (ATR) Indicator, use indicator [Type](#) as `Atr`.

### Exponential Moving Average (EMA)

Moving average Indicators are used to define the direction of the trend. To render a EMA Indicator, use indicator [Type](#) as `Ema`.

### Momentum

Momentum shows the speed at which the price of the stock is changing. To render a Momentum indicator, use indicator [Type](#) as `Momentum`. Momentum indicator will be represented by two lines (upperLine, signalLine). In momentum indicator the upperBand value is always renders at the value 100.

### Moving Average Convergence Divergence (MACD)

MACD is based on the difference between two EMA's. To render a MACD Indicator, use indicator [Type](#) as `Macd`. MACD indicator will be represented by MACD line, signal line, MACD histogram. MACD histogram is used to differentiate MACD line and signal line.

### Relative Strength Index (RSI)

RSI shows how strongly a stock is moving in its current direction. To render a RSI Indicator, use indicator [Type](#) as `Rsi`. RSI indicator will be represented by three lines (upperBand, lowerBand, signalLine). The upperBand and lowerBand values are customized by [OverBought](#) and [OverSold](#) properties of indicator and the signalLine is calculated by RSI formula.

### Simple Moving Average (SMA)

Moving average Indicators are used to define the direction of the trend. To render a SMA Indicator, use indicator [Type](#) as `Sma`.

### Stochastic

It shows how a stock is, when compared to previous state. To render a Stochastic indicator, use indicator [Type](#) as `Stochastic`.

Stochastic indicator will be represented by four lines (upperLine, lowerLine, periodLine, signalLine).

In stochastic indicator the upperBand value and lowerBand value is customized by [OverBought](#) and [OverSold](#) properties of indicators and the periodLine and signalLine is render based on stochastic formula.

### Triangular Moving Average (TMA)

Moving average Indicators are used to define the direction of the trend. To render a TMA Indicator, use indicator [Type](#) as `Tma`.

### Bollinger Band

<!-- markdownlint-disable MD034 -->



A Stock Chart overlay that shows the upper and lower limits of normal price movements based on the standard deviation of prices.

To render a Bollinger Band, use indicator [Type](#) as `BollingerBand`. Bollinger band will be represented by three lines (upperLine, lowerLine, signalLine). (<https://help.syncfusion.com/cr/aspnetcore-js2/Syncfusion.EJ2.Charts.StockChartStockChartIndicator.html#SyncfusionEJ2ChartsStockChartStockChartIndicatorType>) and [StandardDeviations](#) is 2.

### CSHTML

```
<script src="~/financial-data.js"></script>
<div class="control-section">
 <div id="control-container">

@ (Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").ExportType(new List<Object>() { }).TrendlineType(new List<Object>()
{ })
 .Series(sr
=>
 {

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).Name("Apple
Inc").DataSource("data").Add();
 })
 .Render())

 <script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.series[0].indicators = [
 {
 type: 'BollingerBands', field: 'close', seriesName:
'Apple Inc',
 period: 10, upperLine: { color: '#ffb735', width: 1 },
 lowerLine: { color: '#f2ec2f', width: 1 }
 }
]
 }
 </script>
```

### INDICATOR.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

## Tooltip

<!-- markdownlint-disable MD036 -->

Stock Chart will display details about the points through tooltip, when the mouse is moved over the point.

### Default tooltip

By default, tooltip is not visible. Enable the tooltip by setting [Enable](#) property to true .

### CSHTML

```
<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price")

 .Series(sr
=>
 {

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
());

}).Tooltip(tp => tp.Enable(true))

 .Render())

<script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.tooltip = { enable: true };
 }

</script>
```

### TOOLTIP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 {
 public IActionResult Default()
 {
 return View();
 }
 }
 }
}
```

<!-- markdownlint-disable MD013 -->

### Format the tooltip

<!-- markdownlint-disable MD013 -->

By default, tooltip shows information of x and y value in points. In addition to that, you can show more information in tooltip. For example the format `${series.name} ${point.x}` shows series name and point x value.

### CSHTML

```
<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock Price"))

=>
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add();
 sr.Tooltip(tp =>
 {
 tp.Enable(true).Header("Unemployment").Format("${point.x} : ${point.y}");
 })
 })
 .Render();

<script>
 var data = window.chartData;
</script>
```

### FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Position the tooltip

By default, the tooltip is positioned at the left side of the stock chart. You can move the tooltip along with the mouse by setting **Nearest** to the [Position](#) property.

### CSHTML

```
<script src="stock-chart/stockchart-feature/position/financialdata.js"></script>
@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock Price"))

 .Series(sr =>
 {
```

```

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Tooltip(tp => tp.Enable(true).Shared(true).Position("Nearest"))
 .Render()
</script>
var data = window.chartData;
</script>

```

## POSITION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

## Customize the appearance of the tooltip

The [Fill](#) and [Border](#) properties are used to customize the background color and border of the tooltip respectively. The [TextStyle](#) property in the tooltip is used to customize the font of the tooltip text.

## CSHTML

```

<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price")

 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Tooltip(tp =>
tp.Enable(true).Format("${point.x} : ${point.y}").Fill("#7bb4eb"))
 .Render())

<script>
var data = window.chartData;
function stockload(args) {
 args.stockChart.tooltip = {
 border: {
 width: 2,
 color: 'grey'

```

```

 }
 };
}
</script>

```

### CUSTOMTOOLTIP.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### Add Crosshair

Crosshair has a vertical and horizontal line to view the value of the axis at mouse or touch position.

Crosshair lines can be enabled by using [Enable](#) property in the **Crosshair**.

### CSHTML

```

<script src="~/financial-data.js"></script>
@(Html.EJS().StockChart("container").Title("AAPL Stock
Price").ExportType(new List<Object>() { })
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
.Crosshair(cr=>cr.Enable(true))
 .Render())
<script>
 var data = window.chartData;
</script>

```

### CROSSHAIR.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart

```

```
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Tooltip for axis

Tooltip label for an axis can be enabled by using [Enable](#) property of `CrosshairTooltip` in the corresponding axis.

### CSHTML

```
<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").ExportType(new List<Object>() { })
 .PrimaryXAxis(xaxis
=>xaxis.MajorTickLines(mg=>mg.Width(0).Color("transparent")).CrosshairToolti
p(ct => ct.Enable(true)).LineStyle(ls=>ls.Color("transparent")

).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).CrosshairTooltip(ct=>c
t.Enable(true)))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("transparent")).CrosshairTooltip(ct=>
ct.Enable(true))

 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render())

<script>
 var data = window.chartData;
</script>
```

### AXIS-TOOLTIP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

```

 }
}
}

```

### Customization

The [Fill](#) and [TextStyle](#) property of the [CrosshairTooltip](#) is used to customize the background color and font style of the crosshair label respectively. Color and width of the crosshair line can be customized by using the [Line](#) property in the crosshair.

### CSHTML

```

<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").ExportType(new List<Object>() { })

=>
 .Series(sr
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
 ();
 })
 .Render())

<script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.crosshair = { enable: true, line: { width: 2,
color: 'green' } };
 args.stockChart.primaryYAxis = {
 lineStyle: { color: 'transparent' },
 majorTickLines: { color: 'transparent', width: 0 },
 crosshairTooltip: { enable: true, fill: 'green' }
 };
 args.stockChart.primaryXAxis = {
 majorGridLines: { color: 'transparent' },
 crosshairTooltip: { enable: true, fill: 'green' }
 };
 }
</script>

```

### CUSTOM.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

```

 }
}
}

```

### Add Trackball

Trackball is used to track a data point closest to the mouse or touch position. Trackball marker indicates the closest point and trackball tooltip displays the information about the point.

Trackball can be enabled by setting the [Enable](#) property of the crosshair to true and [Shared](#) property in **Tooltip** to true in chart.

### CSHTML

```

@Html.EJS().Chart("container-vertical").Series(series =>
{
 series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("yValue")
 .Marker(ViewBag.marker)
 .DataSource(ViewBag.dataSource).Name("John").Width(2).Add();

 series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("yValue1")
 .Marker(ViewBag.marker)
 .DataSource(ViewBag.dataSource).Name("Andrew").Width(2).Add();

 series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("x").YName("yValue2")
 .Marker(ViewBag.marker)
 .DataSource(ViewBag.dataSource).Name("Thomas").Width(2).Add();})\
 .PrimaryXAxis(px =>
px.EdgeLabelPlacement(Syncfusion.EJ2.Charts.EdgeLabelPlacement.Shift)
 .MajorGridLines(ViewBag.majorGridLines).LineStyle(ViewBag.lineStyle)
 .ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime)
 .Skeleton("y"))
 .PrimaryYAxis(py => py.Title("Revenue")
 .MajorTickLines(ViewBag.majorTickLines)
 .LineStyle(ViewBag.lineStyle)
 .Minimum(10)
 .Maximum(80)
 .LabelFormat("{value}M"))
 .ChartArea(area => area.Border(ViewBag.ChartBorder))
 .Tooltip(tl =>tl.Enable(true).Shared(true))
 .Crosshair(cr =>
cr.Enable(true).LineType(Syncfusion.EJ2.Charts.LineType.Vertical))
 .Title("Average Sales per Person").Render()

```

### TRACKBALL.CS

```

public ActionResult Index()
{
 List<TrackballChartData> chartData = new
List<TrackballChartData>
{
 new TrackballChartData { xValue = new DateTime(2000, 2, 11),
yValue = 14, yValue1 = 39, yValue2 = 60 },
 new TrackballChartData { xValue = new DateTime(2000, 9, 4),
yValue = 20, yValue1 = 30, yValue2 = 55 },

```



```

 new TrackballChartData { xValue = new DateTime(2001, 2, 11),
yValue = 25, yValue1 = 28, yValue2 = 48 },
 new TrackballChartData { xValue = new DateTime(2001, 9, 16),
yValue = 21, yValue1 = 35, yValue2 = 57 },
 new TrackballChartData { xValue = new DateTime(2002, 2, 7),
yValue = 13, yValue1 = 39, yValue2 = 62 },
 new TrackballChartData { xValue = new DateTime(2002, 9, 7),
yValue = 18, yValue1 = 41, yValue2 = 64 },
 new TrackballChartData { xValue = new DateTime(2003, 2, 11),
yValue = 24, yValue1 = 45, yValue2 = 57 },
 new TrackballChartData { xValue = new DateTime(2003, 9, 14),
yValue = 23, yValue1 = 48, yValue2 = 53 },
 new TrackballChartData { xValue = new DateTime(2004, 2, 6),
yValue = 19, yValue1 = 54, yValue2 = 63 },
 new TrackballChartData { xValue = new DateTime(2004, 9, 6),
yValue = 31, yValue1 = 55, yValue2 = 50 },
 new TrackballChartData { xValue = new DateTime(2005, 2, 11),
yValue = 39, yValue1 = 57, yValue2 = 66 },
 new TrackballChartData { xValue = new DateTime(2005, 9, 11),
yValue = 50, yValue1 = 60, yValue2 = 65 },
 new TrackballChartData { xValue = new DateTime(2006, 2, 11),
yValue = 24, yValue1 = 60, yValue2 = 79 },
 };
 ViewBag.dataSource = chartData;
 return View();
}
public class TrackballChartData
{
 public DateTime xValue;
 public double yValue;
 public double yValue1;
 public double yValue2;
}

```

## Legend

Legend provides information about the series rendered in the Stock Chart. Legend can be added to a Stock Chart by enabling the [Visible](#) option in the [LegendSettings](#).

## Position and Alignment

By using the [Position](#) property, legend can be placed at **Left**, **Right**, **Top**, **Bottom** or **Custom** of the Stock Chart. The legend is positioned at the bottom of the Stock Chart, by default.

## CSHTML

```

<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name("China").Add();
 })
)

```

```

 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend => legend.Visible(true).Position("Top"))
 .Render()
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>

```

### POSITION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

[Custom](#) position is used to position the legend anywhere in the Stock Chart using x, y coordinates.

### CSHTML

```

<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name("China").Add();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend =>
legend.Visible(true).Position("Custom").location(location =>
location.x(200).y(20)))
 .Render()
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>

```

**CUSTOM.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

**Legend Alignment**

The legend can be align as **Center**, **Far** or **Near** to the Stock Chart using [Alignment](#) property.

**CSHTML**

```
<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name
("China").Add();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend =>
legend.Visible(true).Position("Bottom").Alignment("Near"))
 .Render())
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>
```

**ALIGNMENT.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
```

```
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Customization

To change the legend icon shape, [LegendShape](#) property in the [Series](#) can be used. By default legend icon shape is `SeriesType`.

### CSHTML

```
<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name("China").legendShape("Pentagon").Add();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend => legend.Visible(true))
 .Render())
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>
```

### CUSTOMIZATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Legend Size

By default, legend takes 20% - 25% of the Stock Chart's height horizontally, when it is placed on top or bottom position and 20% - 25% of the width vertically, while placing on left or right position of the Stock Chart. The default legend size can be changed by using the [Width](#) and [Height](#) property of the `LegendSettings`.

### CSHTML

```
<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name("China").Add();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend =>
legend.Visible(true).width(500).height(50).border(border =>
border.width(1).color("pink")))
 .Render())
<script>
var data = window.chartData;
var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
}
</script>
```

### SIZE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Legend Item Size

The size of the legend items can be customized by using the [ShapeHeight](#) and [ShapeWidth](#) property.

**CSHTML**

```

<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name("China").Add();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend =>
legend.Visible(true).shapeHeight(15).shapeWidth(15))
 .Render())
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>

```

**ITEM-SIZE.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

**Collapsing Legend Item**

By default, series name will be displayed as legend. To skip the legend for a particular series, empty string to the series name can be given.

**CSHTML**

```

<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {

```

```

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend => legend.Visible(true))
 .Render();
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>

```

### COLLAPSING.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### Legend Title

The title for legend can be set using [Title](#) property in [LegendSettings](#). Customize the [FontStyle](#), [Size](#), [FontWeight](#), [Color](#), [TextAlignment](#), [FontFamily](#), [Opacity](#) and [TextOverflow](#) of legend title. [TitlePosition](#) is used to set the legend position in [Top](#), [Left](#) and [Right](#) position. [MaximumTitleWidth](#) is used to set the width of the legend title. By default, it will be 100px.

### CSHTML

```

<script src="~/datasource.js"></script>
@(Html.EJS().StockChart("container").Load("load").Title("Unemployment Rates
1975-2010")
 .PrimaryXAxis(xaxis =>
xaxis.ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime))
 .Series(sr =>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Name("China").Add();
 })
 .Tooltip(tooltip => tooltip.Enable(true))
 .LegendSettings(legend =>
legend.Visible(true).title("Countries").titlePosition("Top"))

```

```

 .Render())
<script>
 var data = window.chartData;
 var load = function (args) {
 args.stockChart.indicatorType = [];
 args.stockChart.trendlineType = [];
 }
</script>

```

### TITLE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### Period selector

The period selector allows to select a range with specified periods. By default the period selector is enabled in stock chart.

#### Periods

<!-- markdownlint-disable MD034 -->

Periods is an array of objects that allows users to specify the range of [Periods]

([https://help.syncfusion.com/cr/aspnetcore-](https://help.syncfusion.com/cr/aspnetcore-js2/Syncfusion.EJ2.Charts.StockChart.html#SyncfusionEJ2ChartsStockChartPeriods)

[js2/Syncfusion.EJ2.Charts.StockChart.html#SyncfusionEJ2ChartsStockChartPeriods](https://help.syncfusion.com/cr/aspnetcore-js2/Syncfusion.EJ2.Charts.StockChart.html#SyncfusionEJ2ChartsStockChartPeriods)). The **Interval** property specifies the count value of the button, and the **Text** property specifies the text to be displayed on button. The **IntervalType** property allows users to customize the intervals of the buttons. The **IntervalType** property supports the following interval types:

- Auto
- Years
- Quarter
- Months
- Weeks
- Days
- Hours
- Minutes
- Seconds



**CSHTML**

```

@ (Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").ExportType(new List<Object>() { }).SeriesType(new List<Object>() {
}).IndicatorType(new List<Object>() { }).TrendlineType(new List<Object>() {
}))
 .PrimaryXAxis(xaxis
=>xaxis.MajorGridLines(mg=>mg.Color("transparent")
))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("transparent").Width(0)
).LineStyle(ls=>ls.Color("transparent")).Crosshair(cr =>
cr.Enable(true)))
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).XName("X").YName("Y").Name("google").DataSource("data").Add();
 })
 .Periods(pr =>
 {
pr.IntervalType(Syncfusion.EJ2.Charts.RangeIntervalType.Minutes).Interval(1)
.Text("1M").Add();

pr.IntervalType(Syncfusion.EJ2.Charts.RangeIntervalType.Minutes).Interval(30)
.Text("30M").Add();

pr.IntervalType(Syncfusion.EJ2.Charts.RangeIntervalType.Hours).Interval(1).Text("1H").Add();

pr.IntervalType(Syncfusion.EJ2.Charts.RangeIntervalType.Hours).Interval(12).Text("12H").Selected(true).Add();
 })
 .Render())

<script>
 var series1 = [];
 var point1;
 var value = 80;
 var i;
 for (i = 1; i < 500; i++) {
 if (Math.random() > .5) {
 value += Math.random();
 } else {
 value -= Math.random();
 }
 point1 = { x: new Date(2000, 1, 1, 0, i), y: value.toFixed(1) };
 series1.push(point1);
 }
 var data = series1;

</script>

```

**PERIOD.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

Visibility of period selector

The [EnablePeriodSelector](#) property allows users to toggle the visibility of period selector.

**CSHTML**

```

<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").EnablePeriodSelector(false)

=>

 .Series(sr
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
();
 })
 .Render())

<script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.crosshair = { enable: true };
 }
</script>

```

**VISIBILITYPERIOD.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

```

 }
}
}

```

### Range selector

The period selector allows to select a range with specified periods. By default the period selector is enabled in stock chart.

### Selecting Range

The left and right thumb of RangeNavigator are used to indicate the selected range in the large collection of data. Following are the ways you can select a range.

- By dragging the thumbs.
- By tapping on the labels.
- By setting the start and end through Date Range button.

Following code example shows the [EnableSelector](#) property allows users to toggle the visibility of enable selector.

### CSHTML

```

<script src="~/financial-data.js"></script>

@(Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").EnableSelector(false)

=>

 .Series(sr
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
());
 })
 .Render())

<script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.crosshair = { enable: true };
 }
</script>

```

### RANGE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

```

 }
}
}

```

## Print and Export

The rendered stock chart can be exported to **JPEG**, **PNG**, **SVG**, or **PDF** format using the export dropdown button in the period selector toolbar. You can choose the required format using the export dropdown button in stock-chart.

The rendered stock chart can be printed directly using print button in period selector toolbar.

### CSHTML

```

<script src="~/financial-data.js"></script>
 @(Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .PrimaryXAxis(xaxis
=>xaxis.MajorGridLines(mg=>mg.Color("transparent")))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("transparent").Width(0))
 .LineStyle(ls=>ls.Color("transparent")))
 .Series(sr
=>
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
 ();
 })
 .Render())

<script>
 var data = window.chartData;
</script>

```

### PRINT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

## Disable Export and print

To empty the value of **ExportType** for to disable the Export button.

### CSHTML

```

<script src="~/financial-data.js"></script>
 @Html.EJS().StockChart("container").Title("AAPL Stock
Price").ExportType(new List<Object>() { })
 .PrimaryXAxis(xaxis
=>xaxis.MajorGridLines(mg=>mg.Color("transparent")))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("transparent").Width(0))
 .LineStyle(ls=>ls.Color("transparent")))
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
());
 })
 .Render()

<script>
 var data = window.chartData;
</script>

```

### DISABLE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

## Appearance

### Stock Chart Title

Stock Chart can be given a title using [Title](#) property, to show the information about the data plotted.

### CSHTML

```

<script src="~/financial-data.js"></script>
 @Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .Series(sr
=>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
());
 })
 .TitleStyle(ts =>

```

```
ts.Color("#E27F2D").FontFamily("Arial").FontStyle("italic").FontWeight("regular").Size("20px"))
 .Render()

<script>
 var data = window.chartData;
</script>
```

**TITLE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

<!-- markdownlint-disable MD036 -->

**Title Customizations**

The **TextStyle** property of chart title provides options to customize the **Size**, **Color**, **FontFamily**, **FontWeight**, **FontStyle**, **Opacity**, **TextAlignment** and **TextOverflow**.

**CSHTML**

```
<script src="~/financial-data.js"></script>
 @Html.EJS().StockChart("container").Title("AAPL Stock Price")
 .Series(sr
=> {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
 ();
 })
 .TitleStyle(ts =>
ts.Color("#E27F2D").FontFamily("Arial").FontStyle("italic").FontWeight("regular").Size("20px").TextOverflow(Syncfusion.EJ2.Charts.TextOverflow.Wrap))
 .Render()

 <script>
 var data = window.chartData;
 </script>
```

**TITLEWRAP.CS**

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}
```

### Stock Chart Theme

Changing theme will affect background color, gridlines, tooltip colors and appearance.

[Theme](#) property of Stock chart is shipped with several built-in themes such as Material, Fabric, Bootstrap, HighContrastLight, MaterialDark, FabricDark, FabricDark, HighContrast and BootstrapDark.

### CSHTML

```
<script src="~/financial-data.js"></script>

@Html.EJS().StockChart("container").Load("stockload").Title("AAPL Stock
Price").Theme(Syncfusion.EJ2.Charts.ChartTheme.HighContrast)

=>

 .Series(sr
 {
 sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Candle).DataSource("data").Add
 ();
 })
 .Render()

<script>
 var data = window.chartData;
 function stockload(args) {
 args.stockChart.titleStyle = {
 size: '20px'
 };
 }
</script>
```

### THEME.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
```

```

 public IActionResult Default()
 {
 return View();
 }
 }
}

```

## See Also

- [Axis Customization](#)

<!-- markdownlint-disable MD036 -->

## Stock Events

Stock Events visualizes stockevents in stockchart. 'SplineSeries' is used to represent selected data value. You can customize the specific data value using **StockEvents** event.

### CSHTML

```

<script src="~/financial-data.js"></script>
@ (Html.EJS().StockChart("container")..SeriesType(new
List<Object>() { }).TrendlineType(new List<Object>() { }).IndicatorType(new
List<Object>() { })
 .PrimaryXAxis(xaxis =>xaxis.MajorGridLines(mg=>mg.Width(0)
).ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).CrosshairTooltip(ct=>c
t.Enable(true)))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("transparent").Width(0)
).LineStyle(ls=>ls.Color("transparent").CrosshairTooltip(ct=>ct.Enable(fals
e)))
 .Series(sr =>
 {
sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).DataSource("data").XNa
me("x").YName("high").Close("high").Add();
 })
 .Crosshair(cr=>cr.Enable(true))
 .ChartArea(area => area.Border(br=>br.Width(0)))
 .StockEvents(se =>
 {
 se.Date(new DateTime(2012, 03,
01)).Text("Q2").Description("2012 Quarter2
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).Add();
 se.Date(new DateTime(2012, 03,
20)).Text("Open").Description("Markets opened").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 06,
01)).Text("Q3").Description("2013 Quarter3
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).

```



```

 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 09,
01)).Text("Q4").Description("2013 Quarter4
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 07,
30)).Text("G").Description("Google stocks bought").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 10,
01)).Text("Y").Description("Yahoo stocks
sold").Type(Syncfusion.EJ2.Charts.FlagType.Square).
 Background("#841391").Border(br =>
br.Color("#841391")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 12,
01)).Text("Y2").Description("Year
2013").Type(Syncfusion.EJ2.Charts.FlagType.Pin).
 Background("#6322e0").ShowOnSeries(false).Border(br =>
br.Color("#6322e0")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2013, 03,
01)).Text("Q2").Description("2013 Quarter2
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2013, 03,
20)).Text("Q2").Description("Surge in
Stocks").Type(Syncfusion.EJ2.Charts.FlagType.ArrowUp).
 Background("#3ab0f9").Border(br =>
br.Color("#3ab0f9")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2013, 06,
01)).Text("Q3").Description("2013 Quarter3
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2013, 09,
01)).Text("Q4").Description("2013 Quarter4
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2013, 12,
0)).Text("Y3").Description("Year
2014").Type(Syncfusion.EJ2.Charts.FlagType.Pin).ShowOnSeries(false).
 Background("#6322e0").Border(br =>
br.Color("#6322e0")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2014, 03,
01)).Text("Q2").Description("2014 Quarter2
starts").Type(Syncfusion.EJ2.Charts.FlagType.ArrowDown).ShowOnSeries(false).
 Background("#3ab0f9").Border(br =>
br.Color("#3ab0f9")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2014, 06,
01)).Text("Q3").Description("2014 Quarter3 starts").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts => ts.Color("white")).Add();

```

```

 se.Date(new DateTime(2014, 09,
01)).Text("Q4").Description("2014 Quarter4
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2014, 12,
0)).Text("Y4").Description("Year
2015").Type(Syncfusion.EJ2.Charts.FlagType.Pin).ShowOnSeries(false).
 Background("#6322e0").Border(br =>
br.Color("#6322e0")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2014, 02,
02)).Text("End").Description("Markets
Closed").Type(Syncfusion.EJ2.Charts.FlagType.ArrowDown).
 Background("#3ab0f9").Border(br =>
br.Color("#3ab0f9")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2015, 01,
07)).Text("A").Description("This is event description").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2015, 01,
02)).Text("Q1").Description("Add longer text").
 Background("#dd3c9f").Border(br =>
br.Color("#dd3c9f")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2015, 02,
12)).Text("Close").Description("Markets closed").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts => ts.Color("white")).Add();
 })
 .Render();

<script>
 var data = window.aapl;
</script>

```

### STOCKEVENTS.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

### Stock Events for individual series

By default, stock events will be showed for all series. Now, you can set the stock events for particular series using `SeriesIndexes` property.

### CSHTML

```
<script src="~/financial-data.js"></script>
 @(Html.EJS().StockChart("container")..SeriesType(new
List<Object>() { }).TrendlineType(new List<Object>() { }).IndicatorType(new
List<Object>() { })
 .PrimaryXAxis(xaxis =>xaxis.MajorGridLines(mg=>mg.Width(0)
) .ValueType(Syncfusion.EJ2.Charts.ValueType.DateTime).CrosshairTooltip(ct=>c
t.Enable(true)))
 .PrimaryYAxis(yaxis
=>yaxis.MajorTickLines(mt=>mt.Color("transparent").Width(0)
) .LineStyle(ls=>ls.Color("transparent").CrosshairTooltip(ct=>ct.Enable(fals
e)))
 .Series(sr =>
 {

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).DataSource("data").XNa
me("x").YName("high").Add();

sr.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Spline).DataSource("data").XNa
me("x").YName("low").Add();
 })
 .Crosshair(cr=>cr.Enable(true))
 .ChartArea(area => area.Border(br=>br.Width(0)))
 .StockEvents(se =>
 {
 se.Date(new DateTime(2012, 03,
01)).Text("Q2").Description("2012 Quarter2
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d").SeriesIndexes([0])).Add();
 se.Date(new DateTime(2012, 03,
20)).Text("Open").Description("Markets opened").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 06,
01)).Text("Q3").Description("2013 Quarter3
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 09,
01)).Text("Q4").Description("2013 Quarter4
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([1]).Add();
 se.Date(new DateTime(2012, 07,
30)).Text("G").Description("Google stocks bought").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([0]).Add();
```

```

 se.Date(new DateTime(2012, 10,
01)).Text("Y").Description("Yahoo stocks
sold").Type(Syncfusion.EJ2.Charts.FlagType.Square).
 Background("#841391").Border(br =>
br.Color("#841391")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2012, 12,
01)).Text("Y2").Description("Year
2013").Type(Syncfusion.EJ2.Charts.FlagType.Pin).
 Background("#6322e0").ShowOnSeries(false).Border(br =>
br.Color("#6322e0")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2013, 03,
01)).Text("Q2").Description("2013 Quarter2
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([0]).Add();
 se.Date(new DateTime(2013, 03,
20)).Text("Q2").Description("Surge in
Stocks").Type(Syncfusion.EJ2.Charts.FlagType.ArrowUp).
 Background("#3ab0f9").Border(br =>
br.Color("#3ab0f9")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([1]).Add();
 se.Date(new DateTime(2013, 06,
01)).Text("Q3").Description("2013 Quarter3
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([0]).Add();
 se.Date(new DateTime(2013, 09,
01)).Text("Q4").Description("2013 Quarter4
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([1]).Add();
 se.Date(new DateTime(2013, 12,
0)).Text("Y3").Description("Year
2014").Type(Syncfusion.EJ2.Charts.FlagType.Pin).ShowOnSeries(false).
 Background("#6322e0").Border(br =>
br.Color("#6322e0")).TextStyle(ts => ts.Color("white")).Add();
 se.Date(new DateTime(2014, 03,
01)).Text("Q2").Description("2014 Quarter2
starts").Type(Syncfusion.EJ2.Charts.FlagType.ArrowDown).ShowOnSeries(false).
 Background("#3ab0f9").Border(br =>
br.Color("#3ab0f9")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([1]).Add();
 se.Date(new DateTime(2014, 06,
01)).Text("Q3").Description("2014 Quarter3 starts").
 Background("#f48a21").Border(br =>
br.Color("#f48a21")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([0]).Add();
 se.Date(new DateTime(2014, 09,
01)).Text("Q4").Description("2014 Quarter4
starts").Type(Syncfusion.EJ2.Charts.FlagType.Flag).
 Background("#6c6d6d").Border(br =>
br.Color("#6c6d6d")).TextStyle(ts =>
ts.Color("white")).SeriesIndexes([1]).Add();

```

```

 se.Date(new DateTime(2014, 12,
0)) .Text("Y4") .Description("Year
2015") .Type(Syncfusion.EJ2.Charts.FlagType.Pin) .ShowOnSeries(false) .
 Background("#6322e0") .Border(br =>
br.Color("#6322e0")) .TextStyle(ts => ts.Color("white")) .Add();
 se.Date(new DateTime(2014, 02,
02)) .Text("End") .Description("Markets
Closed") .Type(Syncfusion.EJ2.Charts.FlagType.ArrowDown) .
 Background("#3ab0f9") .Border(br =>
br.Color("#3ab0f9")) .TextStyle(ts =>
ts.Color("white")) .SeriesIndexes([0]) .Add();
 se.Date(new DateTime(2015, 01,
07)) .Text("A") .Description("This is event description") .
 Background("#f48a21") .Border(br =>
br.Color("#f48a21")) .TextStyle(ts => ts.Color("white")) .Add();
 se.Date(new DateTime(2015, 01,
02)) .Text("Q1") .Description("Add longer text") .
 Background("#dd3c9f") .Border(br =>
br.Color("#dd3c9f")) .TextStyle(ts =>
ts.Color("white")) .SeriesIndexes([0]) .Add();
 se.Date(new DateTime(2015, 02,
12)) .Text("Close") .Description("Markets closed") .
 Background("#f48a21") .Border(br =>
br.Color("#f48a21")) .TextStyle(ts => ts.Color("white")) .Add();
 })
 .Render()

<script>
 var data = window.aapl;
</script>

```

## STOCK-EVENTS-2.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.StockChart
{
 public partial class StockChartController : Controller
 {
 public IActionResult Default()
 {
 return View();
 }
 }
}

```

## Accessibility in ASP.NET MVC Stock chart component

The Stock chart component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Stock chart component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

#### WAI-ARIA attributes

The Stock chart component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Stock chart component:

- img (role)
- button (role)
- region (role)

- `aria-label` (attribute)
- `aria-hidden` (attribute)
- `aria-pressed` (attribute)

### Keyboard interaction

The Stock chart component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Stock chart component.

| **Press** | **To do this** |

| --- | --- |

| **Alt + J** | Moves the focus to the Stock chart element. |

| **Tab** | Moves the focus to the next element in the Stock chart. |

| **Shift + Tab** | Moves the focus to the previous element in the Stock chart. |

| **Down Arrow** | Moves the focus to the data point left side from the selected point. |

| **Up Arrow** | Moves the focus to the data point right side from the selected point. |

| **ESC** | Cancel the tooltip for the data point. |

| **Ctrl + P** | Prints the Stock chart. |

### Ensuring accessibility

The Stock chart component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Stock chart component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Stock chart component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## Switch

### Getting Started with ASP.NET MVC Switch Control

This section briefly explains about how to include [ASP.NET MVC Switch](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.



**~/ LAYOUT.CSHTML**

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

## Add ASP.NET MVC Switch control

Now, add the Syncfusion ASP.NET MVC Switch control in `~/Views/Home/Index.cshtml` page.

**CSHTML**

```
@Html.EJS().Switch("default").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Switch control will be rendered in the default web browser.



## Set text on Switch

This section explains how to set [OnLabel](#) and [OffLabel](#) texts on Switch. In the following example, `onLabel` is set as `ON` and `offLabel` is set as `OFF`.

**CSHTML**

```
@Html.EJS().Switch("default").OnLabel("ON").OffLabel("OFF").Render()
```



**Note:** Switch does not have text support for material themes, and does not support long custom text.

**Note:** [View Sample in GitHub](#).

See also

- [How to customize the switch appearance](#)

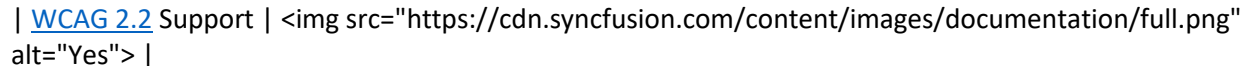
## Accessibility in Switch Button Controls

The Switch component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Switch component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

```

| Screen Reader Support | |

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

The Switch component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Switch component:

| Attributes    | Purpose                                                                                              |
|---------------|------------------------------------------------------------------------------------------------------|
| role          | Indicates the switch component.                                                                      |
| aria-disabled | Indicates that the element is perceivable but disabled, so it is not editable or otherwise operable. |

### Keyboard interaction

The Switch component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Switch component.

| Press | To do this |

| --- | --- |

| Space | When the switch has focus, pressing the Space key changes the state of the switch. |

### Ensuring accessibility

The Switch component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Switch component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Switch component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET Core controls](#)

### Styles and Appearances

To modify the Switch appearance, you need to override the default CSS of Switch component. Find the list of CSS classes and its corresponding section in Switch. Also, you have an option to create your own custom theme for the controls using our [Theme Studio](#).

#### CSS Class | Purpose of Class

|.e-switch-wrapper|.e-switch-inner|To customize the line of the switch in off mode

|.e-switch-wrapper|.e-switch-handle|To customize the handle of the switch in off mode

|.e-switch-wrapper:not(.e-switch-disabled):hover|.e-switch-handle:not(.e-switch-active)|To customize the handle of the switch in off mode when hover

|.e-switch-wrapper:not(.e-switch-disabled):hover|.e-switch-inner:not(.e-switch-active)|To customize the line of the switch in off mode when hover

|.e-switch-wrapper|.e-switch-handle.e-switch-active|To customize the handle of the switch in on mode

|.e-switch-wrapper|.e-switch-on|To customize the line of the switch in on mode

|.e-switch-wrapper:hover|.e-switch-handle.e-switch-active|To customize the handle of the switch in on mode when hover

|.e-switch-wrapper:hover|.e-switch-inner.e-switch-active|.e-switch-on|To customize the line of the switch in on mode when hover

### How To

#### Change Size

The different Switch sizes available are default and small. To reduce the size of default Switch to small, set the [cssClass](#) property to `e-small`.

#### CSHTML

```
<div id='container'>
 <table class='size'>
 <tr>
 <td class='lSize'>Small</td>
 <td>
 @Html.EJS().Switch("switch1").cssClass("e-small").Render()
 </td>
 </tr>
 </table>
</div>
```

```

 </td>
 </tr>
 <tr>
 <td class='lSize'>Default</td>
 <td>
 @Html.EJS().Switch("switch2").Render()
 </td>
 </tr>
 </table>
 </div>
 <style>
 .size tr td {
 padding: 10px;
 }
 .size .lSize {
 font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
 font-size: 13px;
 cursor: pointer;
 user-select: none;
 }
 </style>

```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

### Customize the appearance of a Switch

You can customize the appearance of the Switch component using the CSS rules. Define your own CSS rules according to your requirement and assign the class name to the [cssClass](#) property.

#### Customize Switch bar and handle

Switch bar and handle can be customized as per requirement using CSS rules. Switch bar and handle customized using `cssClass` property. In the following sample, the `border-radius` CSS property for `e-switch-inner` and `e-switch-handle` elements was changed border radius circle to square shape.

### CSHTML

```

@Html.EJS().Switch("switch1").CssClass("square").Render()
@Html.EJS().Switch("switch2").CssClass("custom-switch").Render()
@Html.EJS().Switch("switch3").CssClass("handle-text").Render()
<style>
/* Square Switch */
.e-switch-wrapper.square .e-switch-inner,
.e-switch-wrapper.square .e-switch-handle {
 border-radius: 0;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.custom-switch {
 width: 50px;
 height: 24px;
}

```

```
.e-switch-wrapper.custom-switch .e-switch-handle {
 width: 20px;
 height: 16px;
}
.e-switch-wrapper.custom-switch .e-switch-inner,
.e-switch-wrapper.custom-switch .e-switch-handle {
 border-radius: 0;
}
.e-switch-wrapper.custom-switch .e-switch-handle.e-switch-active {
 left: 42px;
}
/* Customize Handle and Bar Switch */
.e-switch-wrapper.handle-text {
 width: 58px;
 height: 24px;
}
.e-switch-wrapper.handle-text .e-switch-handle {
 width: 26px;
 height: 20px;
 left: 2px;
 background-color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-handle {
 border-radius: 0;
}
.e-switch-wrapper.handle-text .e-switch-handle.e-switch-active {
 left: 46px;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-text:hover .e-switch-inner.e-switch-active .e-switch-on {
 background-color: #4d841d;
 border-color: #4d841d;
}
.e-switch-wrapper.handle-text .e-switch-inner,
.e-switch-wrapper.handle-text .e-switch-off {
 background-color: #e3165b;
 border-color: #e3165b;
}
.e-switch-wrapper.handle-text .e-switch-inner:after,
.e-switch-wrapper.handle-text .e-switch-inner:before {
 font-size: 10px;
 position: absolute;
 line-height: 21px;
 font-family: "Helvetica", sans-serif;
 z-index: 1;
 height: 100%;
 transition: all 200ms cubic-bezier(0.445, 0.05, 0.55, 0.95);
}
.e-switch-wrapper.handle-text .e-switch-inner:before {
 content: "OFF";
 color: #e3165b;
 left: 3px;
}
.e-switch-wrapper.handle-text .e-switch-inner:after {
 content: "ON";
```

```

 right: 5px;
 color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:before {
 color: #fff;
}
.e-switch-wrapper.handle-text .e-switch-inner.e-switch-active:after {
 color: #4d841d;
}
</style>

```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

### Color the Switch

Switch colors can be customized as per the requirement using CSS rules. Switch bar and handle colors customized using `cssClass` property. In the following sample, the `e-switch-inner` and `e-switch-off` elements background and border colors were changed from default colors.

### CSHTML

```

@Html.EJS().Switch("switch1").CssClass("bar-color").Render()
@Html.EJS().Switch("switch2").CssClass("handle-color").Render()
@Html.EJS().Switch("switch3").CssClass("custom-ios").Render()
<style>
/* Custom color Switch */
.e-switch-wrapper.bar-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.bar-color:hover .e-switch-inner.e-switch-active .e-switch-on {
 background-color: #4d841d;
 border-color: #4d841d;
}
.e-switch-wrapper.bar-color .e-switch-inner,
.e-switch-wrapper.bar-color .e-switch-off {
 background-color: #e3165b;
 border-color: #e3165b;
}
.e-switch-wrapper.bar-color .e-switch-handle {
 background-color: #fff;
}
/* handle color Switch */
.e-switch-wrapper.handle-color .e-switch-handle {
 background-color: #e3165b;
}
.e-switch-wrapper.handle-color .e-switch-handle.e-switch-active {
 background-color: #4d841d
}
.e-switch-wrapper.handle-color .e-switch-inner.e-switch-active,
.e-switch-wrapper.handle-color:hover .e-switch-inner.e-switch-active .e-switch-on {
 background-color: #fff;
}

```

```

border-color: #ccc;
}
.e-switch-wrapper.handle-color .e-switch-inner,
.e-switch-wrapper.handle-color .e-switch-off {
 background-color: #fff;
 border-color: #ccc;
}
/* iOS Switch */
.e-switch-wrapper.custom-iOS .e-switch-inner.e-switch-active,
.e-switch-wrapper.custom-iOS:hover .e-switch-inner.e-switch-active .e-
switch-on {
 background-color: #3df865;
 border-color: #3df665;
}
.e-switch-wrapper.custom-iOS {
 width: 42px;
 height: 24px;
}
.e-switch-wrapper.custom-iOS .e-switch-handle {
 width: 20px;
 height: 20px;
}
.e-switch-wrapper.custom-iOS .e-switch-handle.e-switch-active {
 margin-left: -22px;
}
</style>

```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

### Enable ripple for Switch label

By default, label with ripple effect is not available in Switch. You can achieve this using `rippleMouseHandler` method. The following example illustrates how to enable ripple effect for labels in Switch component.

### CSHTML

```

<div id='container'>
 <table class='size'>
 <tr>
 <td class='lSize'><label for='switch1'>USB
Tethering</label></td>
 <td>
 @Html.EJS().Switch("switch1").Render()
 </td>
 </tr>
 </table>
</div>
<style>
.e-switch-wrapper {
 margin-top: 18px;
}

```

```

}
.size tr td {
 padding: 10px;
}
.size .lSize {
 padding-top: 24px;
 font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
 font-size: 13px;
}
.size .lSize label{
 cursor: pointer;
 user-select: none;
}
</style>

```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

### Enable RTL

Switch component has RTL support. This can be achieved by setting [enableRtl](#) as `true`.

The following example illustrates how to enable right-to-left support in Switch component.

### CSHTML

```
@Html.EJS().Switch("default").EnableRtl(true).Render()
```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```

### Set disabled state

Switch can be disabled by setting the [disabled](#) property to true.

The following example illustrates how to disable support in Switch component.

### CSHTML

```
@Html.EJS().Switch("default").Disabled(true).Render()
```

### DEFAULT.CS

```

public ActionResult Default()
{
 return View();
}

```



### Submit name and value in form

The name attribute of the Switch is used to group Switches. When the Switches are grouped in form, the checked items value attribute will post to the server on form submit. The disabled and unchecked Switch values will not be sent to the server on form submit.

In the following code snippet, USB and Wi-Fi in the checked state, and Bluetooth is in disabled state. Values that are in checked state only be sent on form submit.

### CSHTML

```
<div id='container'>
 <form>
 <table class='size'>
 <tr>
 <td class='lSize'>USB</td>
 <td>
 @Html.EJS().Switch("switch1").Checked(true).Render()
 </td>
 </tr>
 <tr>
 <td class='lSize'>Wi-Fi</td>
 <td>
 @Html.EJS().Switch("switch2").Checked(true).Render()
 </td>
 </tr>
 <tr>
 <td class='lSize'>Bluetooth</td>
 <td>
 @Html.EJS().Switch("switch3").Render()
 </td>
 </tr>
 <tr>
 <td>
 @Html.EJS().Button("button").Content("SUBMIT").Render()
 </td>
 </tr>
 </table>
 </form>
</div>
<style>
button {
 margin: 20px 0 0 5px;
}
.size tr td {
 padding: 10px;
}
.size .lSize {
 font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-serif";
 font-size: 13px;
}
.size .lSize label {
 user-select: none;
}
</style>
```

**DEFAULT.CS**

```
public ActionResult Default()
{
 return View();
}
```

## Tabs

### Getting Started with ASP.NET MVC Tab Control

This section briefly explains about how to include [ASP.NET MVC Tab](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

**PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ \_LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Tab control

Now, add the Syncfusion ASP.NET MVC Tab control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Tab("ej2Tab")
 .Items(new List<TabItem> {
 new TabItem { Header = ViewBag.headerText0, Content = "Twitter is an
online social networking service that enables users to send and read short
140-character messages called tweets. Registered users can read and post
tweets, but those who are unregistered can only read them. Users access
Twitter through the website interface, SMS or mobile device app Twitter Inc.
is based in San Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone,
and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, with more than 100 million users posting 340 million
tweets a day in 2012.The service also handled 1.6 billion search queries per
day." },
 new TabItem { Header = ViewBag.headerText1, Content = "Facebook is
an online social networking service headquartered in Menlo Park, California.
Its website was launched on February 4, 2004, by Mark Zuckerberg with his
```

Harvard College roommates and fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had initially limited the website's membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford University. It gradually added support for students at various other universities and later to high-school students." },

```
new TabItem { Header = ViewBag.headerText2, Content = "WhatsApp
Messenger is a proprietary cross-platform instant messaging client for
smartphones that operates under a subscription business model. It uses the
Internet to send text messages, images, video, user location and audio media
messages to other users using standard cellular mobile numbers. As of
February 2016, WhatsApp had a user base of up to one billion,[10] making it
the most globally popular messaging application. WhatsApp Inc., based in
Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion." }
```

```
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .OverflowMode(OverflowMode.Popup)
 .Render()
}
```

```
<style>
```

```
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
}
.container {
 min-width: 350px;
 max-width: 500px;
 margin: 0 auto;
}
```

```
</style>
```

### HOMECONTROLLER.CS

```
public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "Twitter" };
 ViewBag.headerText1 = new TabHeader { Text = "Facebook" };
 ViewBag.headerText2 = new TabHeader { Text = "Whatsapp" };
 return View();
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Tab control will be rendered in the default web browser.

Twitter      Facebook      Whatsapp

Twitter is an online social networking service that enables users to send and read short 140-character messages called tweets. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app. Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day.

Initialize the Tab using JSON items collection

The Tab can be rendered by defining a JSON array. The item is rendered with [header](#) text and [content](#) for each Tab.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@model List<TabTabItem>
@Html.EJS().Tab("ej2Tab").Items(Model).Render()

<style>
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>
```

#### HOMECONTROLLER.CS

```
public ActionResult Index()
{
 List<TabTabItem> items = new List<TabTabItem>();
 items.Add(new TabTabItem { Header = new TabHeader { Text = "Twitter" },
 Content = "Twitter is an online social networking service that enables users to send and read short 140-character messages called 'tweets'. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app. Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day." });
 items.Add(new TabTabItem { Header = new TabHeader { Text = "Facebook" },
 Content = "Facebook is an online social networking service headquartered in Menlo Park, California. Its website was launched on February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The
```

```

founders had initially limited the website's membership to Harvard students,
but later expanded it to colleges in the Boston area, the Ivy League, and
Stanford University. It gradually added support for students at various
other universities and later to high-school students." });
items.Add(new TabBarItem { Header = new TabHeader { Text = "Whatsapp" },
Content = "WhatsApp Messenger is a proprietary cross-platform instant
messaging client for smartphones that operates under a subscription business
model. It uses the Internet to send text messages, images, video, user
location and audio media messages to other users using standard cellular
mobile numbers. As of February 2016, WhatsApp had a user base of up to one
billion,[10] making it the most globally popular messaging application.
WhatsApp Inc., based in Mountain View, California, was acquired by Facebook
Inc. on February 19, 2014, for approximately US$19.3 billion." });
return View(items);
}

```

### Initialize the Tab using HTML elements

The Tab control can be rendered based on the given HTML element using `id` as `target`. Header section must be enclosed with in a wrapper element using `e-tab-header` class and corresponding content must be mapped with `e-content` class.

You need to follow the below structure of HTML elements to render the Tab,

```
`html
```

```

<div id='ej2Tab'> --> Root Tab element
<div class="e-tab-header"> --> Tab header
<div> --> Header Item
</div>
</div>
<div class="e-content"> --> Tab content
<div> --> Content Item
</div>
</div>
</div>
,

```

- Add the HTML template data with its id attribute and add it in your `index.cshtml` file to initialize the Tab.

### CSHTML

```

<div id="ej2Tab">
 <div class="e-tab-header">
 <div>Twitter </div>
 <div>Facebook </div>
 <div>WhatsApp </div>
 </div>

```

```

<div class="e-content">
 <div>
 Twitter is an online social networking service that enables
 users to send and read short 140-character messages called 'tweets'.
 Registered users can read and post tweets, but those who are unregistered
 can only read them. Users access Twitter through the website interface, SMS
 or mobile device app Twitter Inc. is based in San Francisco and has more
 than 25 offices around the world. Twitter was created in March 2006 by Jack
 Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006.
 The service rapidly gained worldwide popularity, with more than 100 million
 users posting 340 million tweets a day in 2012.The service also handled 1.6
 billion search queries per day.
 </div>
 <div>
 Facebook is an online social networking service headquartered in
 Menlo Park, California. Its website was launched on February 4, 2004, by
 Mark Zuckerberg with his Harvard College roommates and fellow students
 Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes.The
 founders had initially limited the website's membership to Harvard students,
 but later expanded it to colleges in the Boston area, the Ivy League, and
 Stanford University. It gradually added support for students at various
 other universities and later to high-school students.
 </div>
 <div>
 WhatsApp Messenger is a proprietary cross-platform instant
 messaging client for smartphones that operates under a subscription business
 model. It uses the Internet to send text messages, images, video, user
 location and audio media messages to other users using standard cellular
 mobile numbers. As of February 2016, WhatsApp had a user base of up to one
 billion,[10] making it the most globally popular messaging application.
 WhatsApp Inc., based in Mountain View, California, was acquired by Facebook
 Inc. on February 19, 2014, for approximately US$19.3 billion.
 </div>
</div>
</div>
<style>
 .e-content .e-item {
 font-size: 12px;
 padding: 10px;
 text-align: justify;
 }
</style>
<script type="text/javascript">
 var tabObj = new ej.navigations.Tab({
 heightAdjustMode: "Auto"
 });
 tabObj.appendTo('#ej2Tab');
</script>

```

**Note:** [View Sample in GitHub.](#)

See also

- [How to load tab with DataSource](#)

## Responsive Modes

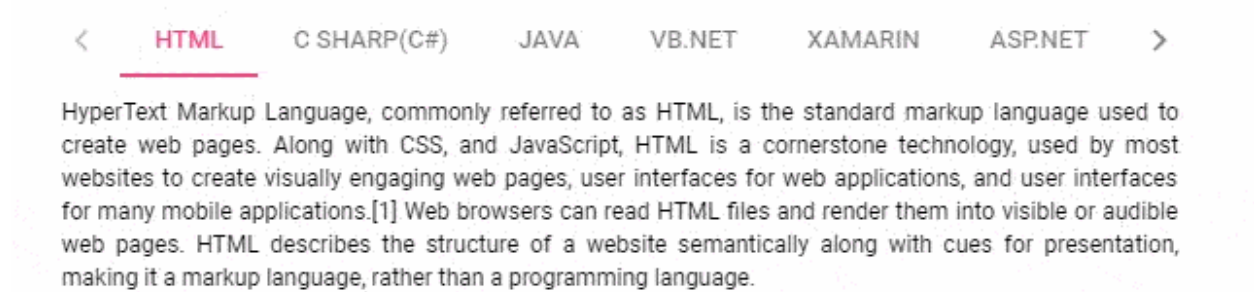
The following section explains about rendering Tab when its width exceeds the viewable area or particularly in a given width. The available modes are as follows:

- Scrollable
- Popup

### Scrollable

The default overflow mode is Scrollable. Scrollable display mode supports displaying the Tab header items in a single line with horizontal scrolling enabled, when the item overflows to the available space.

- The right and left navigation arrow is added at the start and end of the Tab header through which user can navigate towards overflowed items of the Tab header.
- You can also see the overflowed items using touch and swipe action on the header and content section.
- By default, navigation icon in the left direction is disabled, you can see the overflowed items by moving in the right direction.
- By clicking the arrow or by holding the arrow continuously, you can see the overflowed items.



- In devices the navigation icons are not available. You can touch and swipe to see the overflowed items of the Tab header.



### CSHTML

```
@using Syncfusion.EJ2.Navigations;
```



```
@(Html.EJS().Tab("ej2Tab")
.Width("500px")
.Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "HyperText
Markup Language, commonly referred to as HTML, is the standard markup
language used to create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually engaging
web pages, user interfaces for web applications, and user interfaces for
many mobile applications.[1] Web browsers can read HTML files and render
them into visible or audible web pages. HTML describes the structure of a
website semantically along with cues for presentation, making it a markup
language, rather than a programming language." },
 new TabBarItem { Header = ViewBag.headerText1, Content = "C# is
intended to be a simple, modern, general-purpose, object-oriented
programming language. Its development team is led by Anders Hejlsberg. The
most recent version is C# 5.0, which was released on August 15, 2012." },
 new TabBarItem { Header = ViewBag.headerText2, Content = "Java is a
set of computer software and specifications developed by Sun Microsystems,
later acquired by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform computing
environment. Java is used in a wide variety of computing platforms from
embedded devices and mobile phones to enterprise servers and supercomputers.
While less common, Java applets run in secure, sandboxed environments to
provide many features of native applications and can be embedded in HTML
pages." },
 new TabBarItem { Header = ViewBag.headerText3, Content = "The
command-line compiler, VBC.EXE, is installed as part of the freeware .NET
Framework SDK. Mono also includes a command-line VB.NET compiler. The most
recent version is VB 2012, which was released on August 15, 2012." },
 new TabBarItem { Header = ViewBag.headerText4, Content = "Xamarin is
a San Francisco, California based software company created in May 2011[3] by
the engineers that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language Infrastructure (CLI)
and Common Language Specifications (often called Microsoft .NET). With a C#-
shared codebase, developers can use Xamarin tools to write native Android,
iOS, and Windows apps with native user interfaces and share code across
multiple platforms.[5] Xamarin has over 1 million developers in more than
120 countries around the World as of May 2015." },
 new TabBarItem { Header = ViewBag.headerText5, Content = "ASP.NET is
an open-source server-side web application framework designed for web
development to produce dynamic web pages. It was developed by Microsoft to
allow programmers to build dynamic web sites, web applications and web
services. It was first released in January 2002 with version 1.0 of the .NET
Framework, and is the successor to Microsoft's Active Server Pages (ASP)
technology. ASP.NET is built on the Common Language Runtime (CLR), allowing
programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows ASP.NET components to process SOAP
messages." },
 new TabBarItem { Header = ViewBag.headerText6, Content = "The
ASP.NET MVC is a web application framework developed by Microsoft, which
implements the model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is proprietary.
In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET
Web Pages (a platform using only Razor pages) will merge into a unified MVC
6.The project is called ASP.NET vNext." },
 new TabBarItem { Header = ViewBag.headerText7, Content = "JavaScript
(JS) is an interpreted computer programming language. It was originally
```

implemented as part of web browsers so that client-side scripts could interact with the user, control the browser, communicate asynchronously, and alter the document content that was displayed.[5] More recently, however, it has become common in both game development and the creation of desktop applications." }

```
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .OverflowMode(OverflowMode.Scrollable)
 .Render()
}
```

```
<style>
 #container {
 visibility: hidden;
 max-width: 650px;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>
```

### SCROLLABLE.CS

```
public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp(C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
 ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
 ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
 ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
 ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
 return View();
}
```

### Popup

The Popup is the another type of **overflowMode** in which the Tab container holds the items that can be placed within the available space. The rest of the overflowing items for which there is no space to fit within the viewing area are moved to overflow popup container.

- The items placed in popup can be viewed by opening the popup with the help of drop-down icon given at the end of the Tab header.

- If the popup height exceeds the height of the visible area, you can scroll through the popup items and select one.

HTML

C SHARP(C#)

JAVA

VB.NET

XAMARIN



HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Tab("ej2Tab")
 .Width("500px")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "HyperText
Markup Language, commonly referred to as HTML, is the standard markup
language used to create web pages. Along with CSS, and JavaScript, HTML is a
cornerstone technology, used by most websites to create visually engaging
web pages, user interfaces for web applications, and user interfaces for
many mobile applications.[1] Web browsers can read HTML files and render
them into visible or audible web pages. HTML describes the structure of a
website semantically along with cues for presentation, making it a markup
language, rather than a programming language." },
 new TabBarItem { Header = ViewBag.headerText1, Content = "C# is
intended to be a simple, modern, general-purpose, object-oriented
programming language. Its development team is led by Anders Hejlsberg. The
most recent version is C# 5.0, which was released on August 15, 2012." },
 new TabBarItem { Header = ViewBag.headerText2, Content = "Java is a
set of computer software and specifications developed by Sun Microsystems,
later acquired by Oracle Corporation, that provides a system for developing
application software and deploying it in a cross-platform computing
environment. Java is used in a wide variety of computing platforms from
embedded devices and mobile phones to enterprise servers and supercomputers.
While less common, Java applets run in secure, sandboxed environments to
provide many features of native applications and can be embedded in HTML
pages." },
 new TabBarItem { Header = ViewBag.headerText3, Content = "The
command-line compiler, VBC.EXE, is installed as part of the freeware .NET
Framework SDK. Mono also includes a command-line VB.NET compiler. The most
recent version is VB 2012, which was released on August 15, 2012." },
 new TabBarItem { Header = ViewBag.headerText4, Content = "Xamarin is
a San Francisco, California based software company created in May 2011[3] by
the engineers that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language Infrastructure (CLI)
and Common Language Specifications (often called Microsoft .NET). With a C#-
shared codebase, developers can use Xamarin tools to write native Android,
iOS, and Windows apps with native user interfaces and share code across
```

```

multiple platforms.[5] Xamarin has over 1 million developers in more than
120 countries around the World as of May 2015." },
 new TabBarItem { Header = ViewBag.headerText5, Content = "ASP.NET is
an open-source server-side web application framework designed for web
development to produce dynamic web pages. It was developed by Microsoft to
allow programmers to build dynamic web sites, web applications and web
services. It was first released in January 2002 with version 1.0 of the .NET
Framework, and is the successor to Microsoft's Active Server Pages (ASP)
technology. ASP.NET is built on the Common Language Runtime (CLR), allowing
programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows ASP.NET components to process SOAP
messages." },
 new TabBarItem { Header = ViewBag.headerText6, Content = "The
ASP.NET MVC is a web application framework developed by Microsoft, which
implements the model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is proprietary.
In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET
Web Pages (a platform using only Razor pages) will merge into a unified MVC
6.The project is called ASP.NET vNext." },
 new TabBarItem { Header = ViewBag.headerText7, Content = "JavaScript
(JS) is an interpreted computer programming language. It was originally
implemented as part of web browsers so that client-side scripts could
interact with the user, control the browser, communicate asynchronously, and
alter the document content that was displayed.[5] More recently, however, it
has become common in both game development and the creation of desktop
applications." }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .OverflowMode(OverflowMode.Popup)
 .Render()
)
<style>
 #container {
 visibility: hidden;
 max-width: 650px;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>

```

### POPUP.CS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
}

```

```

ViewBag.headerText1 = new TabHeader { Text = "C Sharp (C#)" };
ViewBag.headerText2 = new TabHeader { Text = "Java" };
ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
return View();
}

```

See Also

- [How to prevent content swipe selection](#)
- [Collapsible Tab](#)

## Header

This section explains about modifying the style of Tab header, and configuring its icons and positions.

### Styles

You can customize header styles by adding predefined classes in the Tab root element. The pre-defined CSS class names are as follows:

- **e-fill:** The Selected Tab header background is set as solid fill.
- **e-background:** Tab header has a solid fill background, and the selected header has a highlighted border.
- **e-background e-accent:** Tab header has a solid fill background, and the selected header has a highlighted border with accent color.

**Note:** If the above custom style classes are not included in the root element, the default style is applied to the Tab items.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class='row'>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Header Style </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">

@Html.EJS().DropDownList("styles").Width("100%").DataSource(ViewBag.stylesData).Value("Default").Change("changeStyles").Render()
 </div>
</div>
@ (Html.EJS().Tab("ej2Tab")
 .Width("650px")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "Twitter is an online social networking service that enables users to send and read short 140-character messages called tweets. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app Twitter Inc. is based in San Francisco and has more than 25 offices around the world."

```

```

Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone,
and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, with more than 100 million users posting 340 million
tweets a day in 2012.The service also handled 1.6 billion search queries per
day." },
 new TabBarItem { Header = ViewBag.headerText1, Content = "Facebook
is an online social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College roommates and fellow students Eduardo Saverin,
Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders had
initially limited the website's membership to Harvard students, but later
expanded it to colleges in the Boston area, the Ivy League, and Stanford
University. It gradually added support for students at various other
universities and later to high-school students." },
 new TabBarItem { Header = ViewBag.headerText2, Content = "WhatsApp
Messenger is a proprietary cross-platform instant messaging client for
smartphones that operates under a subscription business model. It uses the
Internet to send text messages, images, video, user location and audio media
messages to other users using standard cellular mobile numbers. As of
February 2016, WhatsApp had a user base of up to one billion,[10] making it
the most globally popular messaging application. WhatsApp Inc., based in
Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion." }
 })
 .Created("tabCreated")
 .Render()
)
<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>
<script type="text/javascript">
 var tabObj;
 function tabCreated() {
 tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 }
 // Change event function for DropDownList component
 function changeStyles(e) {
 removeStyleClass();
 let name = e.itemData;
 if (e.itemData != null && name.value === 'Fill') {
 tabObj.element.classList.add('e-fill');
 } else if (e.itemData != null && name.value === 'Background') {

```

```

 tabObj.element.classList.add('e-background');
 } else if (e.itemData != null && name.value === 'Accent') {
 tabObj.element.classList.add('e-background');
 tabObj.element.classList.add('e-accent');
 }
}
function removeStyleClass() {
 tabObj.element.classList.remove('e-fill');
 tabObj.element.classList.remove('e-background');
 tabObj.element.classList.remove('e-accent');
}
</script>

```

## STYLES.CS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "Twitter" };
 ViewBag.headerText1 = new TabHeader { Text = "Facebook" };
 ViewBag.headerText2 = new TabHeader { Text = "Whatsapp" };
 ViewBag.stylesData = new string[] { "Default", "Fill", "Background",
 "Accent" };
 return View();
}

```

## Icon positions

You can customize the position of the Tab header icons using the icon position property. This property depends on the header items icon CSS property. By default, Tab header icon is placed on left position. The position values are as follows:

- **Left:** Icon is placed on the left of the Tab header item.
- **Right:** Icon is placed on the right of the Tab header item.
- **Top:** Icon is placed on the top of the Tab header item.
- **Bottom:** Icon is placed on the bottom of the Tab header item.

## CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class='row'>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Icon Positions </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">

@Html.EJS().DropDownList("position").DataSource(ViewBag.positionData).Value(
"left").Change("changePositions").Render()
 </div>
</div>
@ (Html.EJS().Tab("ej2Tab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerText0, Content = "Twitter is
an online social networking service that enables users to send and read
short 140-character messages called tweets. Registered users can read and
post tweets, but those who are unregistered can only read them. Users access

```

Twitter through the website interface, SMS or mobile device app Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day." },

new TabBarItem { Header = ViewBag.headerText1, Content = "Facebook is an online social networking service headquartered in Menlo Park, California. Its website was launched on February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had initially limited the website's membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford University. It gradually added support for students at various other universities and later to high-school students." },

new TabBarItem { Header = ViewBag.headerText2, Content = "WhatsApp Messenger is a proprietary cross-platform instant messaging client for smartphones that operates under a subscription business model. It uses the Internet to send text messages, images, video, user location and audio media messages to other users using standard cellular mobile numbers. As of February 2016, WhatsApp had a user base of up to one billion, [10] making it the most globally popular messaging application. WhatsApp Inc., based in Mountain View, California, was acquired by Facebook Inc. on February 19, 2014, for approximately US\$19.3 billion." }

}}

.Render()

)

<style>

```
.e-content .e-item {
 font-size: 12px;
 padding: 10px;
 text-align: justify;
}
```

```
@font-face {
 font-family: 'Socialicons';
 src: url(data:application/x-font-ttf;charset=utf-
```

```
8;base64,AAEAAAKAIAAAAwAgTlMvMvltCfsAAAEoAAAVmNtYXCnKKeOAAABrAAAAEhnbHlml19
XagAAAgwAABhQaGVhZA8dCeEAAADQAAAAANmhoZWEIUQQMAAAArAAAACRobXR4LAAAAAAAYAAAA
sbG9jYR3AIwwAAAH0AAAAGG1heHABIAIAAAABCAAAACBuYW1l0X1q/wAAGlwAAAJVcG9zdGX5D00
AABY0AAAAkwABAAAEAAAAAFWEAAAAAAD9AABAAAAAAAAAAAAAAAAAAACwABAAAAAQAA+iTiP18
PPPUACwQAAAAAANYFYngAAAAAlgVieAAAAAAD9AP0AAAACAACAAAAAAAAAAAEAAAAALafQACwAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABApwCnCQQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAAAaAAAAaAAABQAAwABAAAAFAA
EADQAAAAEAAQAAQAApwn//wAApWd//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoAAAAAiQ
CzgOMBU4F/gZyB9QIcAo+DCgABAAAAAAD0gPzAFUA4gF3AfMAAAEzHwYHFQ8EFR8IPwUfBRUPCCM
vFj0BPwoXNw8fhQEfdhUPAT8CHwkzPyA9Ai8iDwIFHwcPIysBLWYjDwI/AS8PNT8oHx4BDxAdAR8
PHQEHPwE7AR8EMz8dNS8kiw8FAYkFEgQDAyQDAQECAyIBAQMSegkUCw4vBQQFChsGBQdqAgIBAwM
DCAoMDA0NBgYPEA8PFxYVFBQTEhITEREPDgwkCQEQEBQICBAQFChMJBQUFBTURDxAPDw8ODg4NDQw
MDAsLCgkJCQgHBwCFBQwEAWICAQEDAgQEBgYHBwkJCgsOAgEmiwMEBAQUFRQVFRQVFRQVFRQVFRQV
VDw4ODg0NDAwLCwoKCQkICAcGBgUEBAQCAgICAgMEBAYGBgcICQkJCgsLCwwNDQ0ODg4PDw8QEBA
QEBEREREQEQHcBgUEBAICAQEBAQEDAwQFBQYHCAgICgoLCwwNDQ4ODxAREhISEhMTEExMUEXQUFRQ
VGxsaGgcIBwfxNgEBAQ8KCgoIBwCGBQUDAwIBAQECAwMDBQUBGcHCAgICgkKCsMDAwNDg0ODw8
PEBAQEhISEhISEhIREREREREQERAQDw8PDw8ODQ0NDQwLDAoKCgkJCACh/aQEB0cGhgWFBIRDgw
LCAcEAWICAwMFBQYHBwgJCgoLAgE9+AYFBSMeHx4fIB8fFhQUFBQSExIRERAQEA8ODhAQDQ0LCQg
HBgQCAgICBAQEBgYGCAGJCQoLCwwMDQ4ODg8PEBAREBESEhISEXITGBcZGRgZGBgXAU4CAGMEXAK
```



|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>FBAQFBCQCAwMGGhcKFQkMIwIBAwOYAwIBKQECBSgFBgULCgkHBgMBAQEDAwcICgwMDg8PEhITFBuWGBgSeYyYJCAgIBwcHDBEGAgEBAagEBAQGBgYHCAgJCgkLCwsMDAwNDQ4ODg8PDw8QEBAQERITEhESEREREPEA8QDhMEBASFNgEBAQEJCAcGBQMQAQEDAwUGCAgHCAgJCQoKCwsMDA0NDQ4ODg8ODxAPEA8QEBAQEBAQEBIQERAQDw8ODg0NDQwMCwoKCQkICAcGBgUFAwMDAQEBAQEC6RISExISExISEhMSEhESERERERAQEA8QDg8NDg0MDAwLCwoJCQcHBgUEAwICAgIEBggJAwECVNcFBAQVEBEREhESEhITExMTFBMUEhEREBEQEBApDw8PDg4ODQ0MDAwLCwoKCgkICAcHBwUGBQQDAgIBAQEBAgIDBAQFBQcGCAGICQoKCgsMDA0NDQ4PDw8QEBEBBwgIEhMUFhcYGRscHR8fIiIjFBMTehMSEhIREhEREBEQEAMEAwt1YQINCAyEAgIEBAUGBgICQoKDAwNDg8PERUVFhcWGBcYGBkZGRkaGxoTEhISEhEREBAQDw8ODg4MDQwLCgoKCQgHBwYFBQMDAwEBagMFBQgIAAAAAAEEAAAAAzOD9ACWAAATDwYVERUFHTsBPw49AS8OIy8PNSEzPw4vDyE9AS8ODwblCAYFBAQCAgECAwMEBQUGBwgICQoKCwMDA0NDQ0ODg4PDw8QEBDQCgsKCQkJCAGHBwUEBAICAgIEBAUHBwgICQkJCgsK0QoKCgoJCAGIBwcFBAMDAQEBKQoJCggJCAGHBwUFBAQCAQEBAQIEBAUFBwgHCAkJCQkK/tcCAGMFBQYHCAkICQoJCwoLCwoJCQkIA9AJCgsKDAwMDf4LExMTEhESEREQEBApDw4PDQ4MDAoKCQgIBgYEBAMCAgEBAwQFBwcJCQoKCwsMDA0NDAsMCwoKCQkHBwUEAwEBAQEDBAUGCAgJCgoLCwwMDVgCAwMFBgcICQkJCgsLCwwLDAsKCgoJCAGHBgUEAgIBtQ0MDAsLCgoJCQcHBQQDAQEBAQMEBQYIAAABAAAAAAP0A90AqAAAAT8DMx8MHQEPDSsBLxoPCBc/Ah8LEx8PMz8dLw8jDw4CSAoTEhIRCAcGBwUFBQQDAwICAgEDBQoPExYWFAsLCgQFBAUFBAUFBUJJCQkJCyQEBQUGBwcICAKKCgsLDQwODQ8RERQfI5IvNRMGBwYGBwYGBgYGDAsMWAcICAgICQkJCQkKCgoLCgsJExITFBQVFRYWFIMkJTEWFRQREQ8NDAsJBwYFAwEBAQEDBAUHBwkJCwwNDhAQEGsYGBYWFbQSEhAQDg4MCwsC2wQGBQIBAgICAwQEBQUHBgcICBMSDxEdIiUoIXsOCgcCAQIEBAYICBUbHyU37xQTERAPDQwKCQgGBQMQAQEDBgLDhofkUInCwIBAgQEBwcJCwscISf+pBYUEXIQDg4LCwkIBgUDAgEDBACJDA0REhQXJysxRiEhIB4eHRwbGhkZFxcVFRoXfHqTERAODQwKCAcGBAMBAQMFBwkMDQ8RExUXGhsdAAUAAAAA/ED9ABCAKoA6wESAYQAAEdAQ8NKwEvDjU/EB8OJR0BHW8hPw8TLwMhHwUVDxEvEzU/CSchDwMFFR8PPw8vDw8OAR8HFQ8JIY8GPQI/BjMlHQEPBC8DNS8DDwMVDwIjLwM9Ai8BIw8EFQ8DIY8CNw8KFxUFASUzPwgZHWkhPwI1LxALDwICkAMDBQcHCQkKDAwMDQ4ODwMCwsLCgoJDwsJCAYFAGCAwQGBgcICQkKCwsMDA0LCw8ODg4MDAsKCQkHBgQEAv1/AQMFBGkJDAwODwgRERITEwJpExMSEhEQDw4MDAUJBwYEAgEBAgEF/uYOCwkGBAICBAYHCQsMDg8QERETExQTFRQVFBQUBMTehAPDg0LCQgGBAMCAgEBAwMEBQYHCAKc/uoFAGEBASwBAwUGCQoLDQ0PERESEhQUFBQTEhEQEA4MDAKJBgUDAQEDBQcICgsNDg8QERISFBQUBMSERAQDgwMCggHBAMCPAYGBgQEAWEBABQEBAMEBQFbmgHBgYEBAMCAwMEBQUFBjX90AECBAUUBQEBAQEBAgIRAgIBAQIFEAKDAwECBAQEBAQDwICAwUWAwIBAQQQDwWLCQgFAwEBAQEEATEEBBYUFYRYWFxcYFxcXfXyVFBgFBQYBjwYCAgICBAYHCQoLDA4ODxAIERIR/d8FAQIB9wcHDg0NDAwLCgkIBwYFBAICAgQEBQYGDQwODg8REBINdQwMCwsKCgkIBwcGBAQCAQEBAgQFBggICgoLDQ0NDg9929sUEXISERAPDgwMBQkHBgQCAQMFbgkJDAwODwgQERITEwHBBgMBARYXfxcXfxcWfHUUFBMRQ8ODAsJCAYEAWIBAgQFBwkKDA0PDxERExQPEA8PDw8PDw8ODw4ODg4OAEBAQKPCgoUEhIREBANDQsKCAcFAwEBAwUHCAoLDQ4PEBESExQUFBMTehEQDw4NCwoIBwUDAQEDBQcICgsNDg8QEHITEwGSAQICBAUFBgdsBQQFBAQDAgIBAQECAGQFBQYHawCHBgUDAgEBR2h1CAMCAQEBAgIF5wMCAQEBAQED6gUCAQEDAwbbBQICAQIDAWMG0ggEAQICAgTKAQ0OEBASEhQVEiRdAgIBAQITDg0JCAYDAQQFBwoMDhQCAQEBAQNuJBIRERAPDg4NCwoJCAMFBAEBAQIEAAAAAAMAAAAA/QD3QADAFcAlwAANzMRIwUVIzclIXeZET8OHw8RMxEvGw8MAR8PPw41Lw8PDhnW1gIjAQHW1gIDBQgKCwCHBgJCQoKCw4NDAsKCAgHBwUEBAICAQHWAQICAgQDBQUFBgYHBwcJCAkJCgoKCwsLDBgZGhQUEREpdg0MCwoJCQ79xAEBawMFBgYHCAkKCwsMDA4PDQwLCwoJCQcGBgUDAwIBAQMEBAYGCAGJCgoLDQwODQ0MDAoKCQkHBwYEBAMBIGKFWwICW/17AXcUDA0ODgwGBQUEBAMCAQEBAgMFBQcICgoLDA0NDw8Q/qcBhBIREBAPDw4NDQwMCwoKCQkICAcGBgUFBAMGAWEBAGMEBgYHCAgICQkSARIMCwsKCgkICAgGBQUEAwEBAQEDBAUFBgICAKKCgsLDAsLCwsJCggIBwYGBAQDAQEBAQMEBAYGBwgICgkLCwsAAAAABAAAAAPuA/QARgAAExEVHwYhESM1MzU/DzMVIw8GFTMVIxEhPwYRLwYhDwYSAGQFBwgKCgHPb24BAwMGBggJCgsMDQ0ODwgPlUcLCwkIBgQDe3sBBQoKCAcFBAICBAUHCAoK/IUKCgkHBwQDA7v8igYLCgkIBgQDAZuFUBAQDw4ODQwLCgkIBwUEAgGFAwQHCAkKDDOF/mUDBAYICQoLA4ILCgkIBgQDAQQFBwgKCwAAAAAGAAAAAP0A/QAOABEAIAABBQEqAUWAAAEPCR0BHW07AT8NPQEvCCMPASUVMxUjFSM1IzUzNSUPBRUFDTsBPww1Lw4jDwU3ByMfCA8PHw4dAQ8OLw8/DS8FPwIHIIY8NPQE/DwEVHw8hPw8RITLw8hDw4BCgMTCwsFBAQEAgICAwQGBgcICgoLDAwODg8NDQwLCgkICAYGBQQDAWEBAQIDBAgMDiYRNw0B9nR0TXNz/kAFawMDAQIBAgMDBAQGBgcICQkKDAAsIBwCHBwYFBQYFAwMBAQECAwMEBQYGBwgJCQoLDAcIBwCHBwX+MTAQDggIAwICAQEBAQEDAwMICgsMDAsGAgEBAQECAwYiGQoFCQcDagIBAwQFCAGLDA0PERITFRYYFRISEA8NDAsKCAcGBAMCAQEBAwUHCQsOERMUFB0xCACDAWEBAQIFGQ4ODQ0LCgoICAcFBQQCAGMDBGcICgWICBESEhESEBD+pwEDBQYJCgsNDg8IEBISExQCahQTEhIREA8ODQsGCQcGBAL8GAED5gIDBgICgsNDg4QCBIRExP91hMTEhEXEAA4ODQsKCAcGAWFKAQkHCAYGBggICQkKCgkICAgHBgYFBQMDAgIBAgMDBAUFBgYHBwcICQgHBwYGBgYLCwwcBQPYck9yck5zZwYGBwCHDxELCgWLCwsKCgkJBwUFAwECAwMDBAUHBwCIBw0QCwwLDAsMCgoKCAcGBAMBAgIDAQwQFLRkQDwwPCAgJCgoLCQkICAgNDAsKCQwJBQYGBQYEBACbFQsGDA4HCAgJCQ4NDQwNCwwKCgg</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

```

IBgYDAwEBAgMdBQYGBwgJCAoJCgsKCwUMDAwMDAsKCQYFBQUKDAYHCAgJBw0BAGQEBQcHCAkJCgo
KCwsLDQ4NDQ0MDAsGBgkIBQQCAQH+EAoKExMSERAQDQ0LBgkHBgQCAQMFBgkKCw0NEAgQEhITFAJ
HKxQSEhIQDw8NDAsJBQcFBAIBAwQHCAkLDA0PDxASEhIAAAAAAgAAAAAD7gP0AEEAhAAAAARUzFSM
RHws/BxUPAy8OESM1Pw81ER8OMyEzPw4RLw4jISMPDQIbysoDBgUICgYHCAgJCgsLDQ4PEBESE0Q
tICIiEREQDw8ODQWKCgcHBANuGBkVDw4ODgYFBgUEBAMCAv5fAQECawQEBQUGBwCHCAgJCAM0CAk
ICACHBwYFBQQEAWIBAQEBAGMEBAUFBgCHBwgICQj8zAgJCAgHBwCGBQUEBAMCAQON0H/+9BIMCAk
HBAMDAgEBAQEBAgMdBQYHeA4GAwEBAgIDBAUFBgJCwsNDxABVGwKDXANDxEUCwWMDQ00DxAQEvz
CCQgICACHBwYGBAUDAwICAgIDAwUEBgYHBwCICAgJAz4JCAgIBwCHBgYEBQMDAgICAgMdBQGBgC
HBwgICAAAAgAAAAAD7APzAPgBqAAAA8LFQ8MIy8QKwEPDh8bHQEPFi8WPQE/DTMfeJm/Di8ePQE
/Fh8CBR8HDwMfhjSBPwIfBzM/HTUvBz8CPQEvHiMPAi8HIw8dAnALFhMSDw4LCQgFBAIBAgIDAww
FBgUGBgGCawLCQgHChQLCwsHBwkJCgsNDQwMCwsJCgGIBwYFBAMDAQEBAgMEBQcHCRMTdxojFhQ
TEA8OCwUFAwQCAwEBAgIEBQUHCAgKCgWMDg4PEBEREHMTFBUZGBYWFRMSEgSLCwoJCQgIBwYFBQM
CAGACAgMDBAUFBQYGBgYHCAsLCgkIBwCMBwCHBwoKDacPERMZDQ0MDAsKCQgHBgUEAwEBAQICAgM
EBAsMDQ8bTSIfGxkMCwsKCQgIBwYFBQMCAGICBAQGBggICQoLDA0NDw8PERERExIUHxwb/bsBAgM
EBQcHCQUADAQEBAQMFBQYICakLCwwNDg8QEBESEhMUFBWFRcWGBcYGBYWFRUPDxAQEBEREQ4ODg0
NDQ0MDAwMCwoLCgkJCQgHBwCGBgQFAwMDAgEBAQIDBAUGBgQEAgIDBAUHBwkJCgWMDQ4PDxERERM
TFBQVFRYWFxcYGBgUFRQTEBESEhITFBMODg4NDQ0NDA0LDAsKCwoJCQkIBwCHBgYEBQMDAwIBAzC
ECAoLDAwNDQ4NDg0NBgYGBQYKBQQDAwICAQECBAUHDSEODQoEBAMCAgIBAQICAwMEBQUFBQYGBgY
GCACHBgYFBQUIBx0GDAgJCgsNDg8JCAkKCgsLCwwPDg00DQwMDAsLCgoICAgHBgUEBAMCAQEBAgI
EBQYICAYIBwkJCQoKCwsLCgsLCgoHBgYGBQUFBQQEAWMCAQEBAgUGCAkLGg0LCgkICAYDBAMCAQI
DBAQFBgYGBwCIBwkIDQcFBgUEBQgIBgYHEgkJCgoHBgCICakJCgoLDAwMDg0NDQ0MDAsLCgoKCQg
IBwYGBQQEAWMBAQEBAwRbEhMSEREREBAxFxgYGBgYFxcWFhUVFBQTExEREQ8PDg0MDAoKCAcHBQQ
DAgICAwCGBgUDAwEBAQIDAwMFBAYGBwCHCAkJCQoLCgsMDAwMDQ0NDQ4ODhAQEA8PDw4OGB0ZGhg
YFxfGWFxUWFRQUEXISERAQDw4NDAsLCQkHBgUFAwEBAgIDCggHBgUDAgEBAgMDAwUEBgYGCACICQk
JCgoLCwwLDQwNDQ0NDg4AAAAACwAAAAAD8wOYABEAMwBbAKYAYwDTARcBOQFjAZgBoQAAQ8DMzc
vBisBDwEnDwIdAh8FOWE/BjUvBisBDwEnFwcfBDM/Bic1MxUnNw8GIy8HNyClHwsVIxUfBjsBPwY
1MxcVDw0vCzU/CycVPwMfCR0CDwgjLwQPAREjFSMVIzUjNTcPCxUfDyE/DzUvDiMhIw8BJR8DFQ8
GKwEvBjU/Bx8DFR8KPwUHMzUjFQ8GKwEvBjUjDwcdAR8LOWE/CTUvDg8DFTM1NyMHJyMDIgQDAgJ
CAgECAwQFBQYGCgUG2QQDAgIDBAUFBgYGBQYEBQICAQECAGUEBgUGBgUF6wEBawUCAwMEBwGEAgE
BAQFQOEQEDBAYMDhAQDwgGBgYFagIEAQECHQoLCgkJCAcFBAMCAxCAQMDBQQFBhAGBQQEAgIBMwM
BAQMCBAQMBwCICAgIDxAPDg8HBgYEAwMBAQEAGQGBggJCQkJC+UQDg4NDQUGBwCFBAQCAgIDAwU
GBgcICQoLBQwNFAQ50VJFRyAPDQ0LCwkJBwMFAwIBAgQGBwkJCwsNDQ8PDxARAqAQEQ8PDw0NCws
KCAcDBQMCAQIEBgCICgsLDQ0PDxAQEPlgERAPAYoEAgIBAQICBAUGBQYGBgUFBaICAQECAGQFBQU
GBgUGdgEEAgQEBQGBBgIBwCGBgoKAUw7AQEDAwQEBQUFBQQEAWIBAUC+CggGBgMCAgICBAMECgY
HCQsLCwwMCwoSBwCHCAUEAgEBAgIEAwQHwBgJCgsMDg8NDMPKV1AuLVEBbAMEBB8bBQMEAwICAQE
CCAMDAwOAAwMDAgICAQECAGIDAwOAAwMDAwICAQECF4kgBQUCAQECBAQEAIQDCJrYARsEBAMHBQQ
CAQICAwQEBRoPmw0BAQIDAwQFBgYMDBgvmgYEAwIDAQEBAQMCAwQEGREGBgUFBQQECQQEAWICAQE
BAwYHBQYGBwgJCgPDDwwLCggHBgYDAwMBQFQBHQBBQBAQICAwUFBQYHBwhwCgkJCAgGBQQDAQEBAU
LEgEBIib+/yVJBQUGBwCICQkFCgsK9gsLCgoJCQgHBwYFBQMDAQEBAQMDBQUGBwCICQkFCgoL9wo
LCgoJCQgHBwYFBQMDAgID9wQEBAV3BAUDBAMCAQECAwQEBAR3BQQEBAMCAQEBAQJ3GAwRBAUEAwM
DAQEBAQEBAwCKEuCvAwMDAgMBAQEBAWIDAwOvAgUHCAoKDA00QxgOBwCgCgQEAgMCAQICBQQFBws
KDxZTCgkHBgYGBQYFBQMDAgEBAQIEPayslG9vAAAAABIA3gABAAAAAEEEEAAAAAABAAAAAABAAAs
AAQABAAAAAAACAAcADAABAAAAAADAAsAEwABAAAAAAEEAAsAHgABAAAAAAFAAsAKQABAAAAAA
GAAsANAABAAAAAAKACwAPwABAAAAAALABIAawADAAEEECQAAAAIAfQADAAEEECQABABYAfwADAAE
ECQACAA4AlQADAAEEECQADABYAowADAAEEECQAEABYAuQADAAEEECQAFABYAzwADAAEEECQAGABYA5QA
DAAEEECQAKAFgA+wADAAEEECQALACQBUyBTb2NpYWxpY29uc1JlZ3VsYXJTB2NpYWxpY29uc1NvY21
hbGljb25zVmVyc2lvbiAxLjBTb2NpYWxpY29uc0ZvbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmNmdXN
pb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAUwBvAGMAAQBhAGwAaQBjAG8AbgB
zAFIAZQBnAHUAbhAHIAUwBvAGMAAQBhAGwAaQBjAG8AbgBzAFMAbwBjAGkAYQBzAGkAYwBvAG4
AcwBWAGUAcgBzAGkAbwBuACAAMQAuADAAUwBvAGMAAQBhAGwAaQBjAG8AbgBzAEYAbwBuAHQAIA
nAGUAbgBlAHIAIYQB0AGUAZAAGAHUAcwBpAG4AZwAgAFMAEQBuAGMAZgBlAHMAAQBvAG4AIAABNAGU
AdABYAG8AIABTahQAdQBkAGkAbwB3AHcAdwAuAHMAEQBuAGMAZgBlAHMAAQBvAG4ALgBjAG8AbQQA
AAAACAAAAAAsAAAAAAsAAAAAAsAAAAAAsBAgEDAQQBBQEGAQCBCAEJAQoBCwE
MAAh3aGF0c2FwcAd0d2l0dGVyBXZpbWVvCWLuc3RhZ3JhbQhsaW5rZWRpbghmYWNlYm9vawtnb29
nbGUtcGxlcwZ0dWlibHIIc2t5cGUtMDEIeW9ldHVizTEAAAA=) format('trueType');
font-weight: normal;
font-style: normal;

```

```

 }
 .e-tab-icon {
 font-family: 'Socialicons' !important;
 }
 .e-icons.e-tab-icon {
 position: relative;
 top: 1px;
 }
 .e-twitter:before {
 content: '\a701';
 }
 .e-facebook:before {
 content: '\a705';
 }
 .e-whatsapp:before {
 content: '\a700';
 }
}
</style>
<script type="text/javascript">
 function changePositions(e) {
 var tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 for (var i = 0; i < tabObj.items.length; i++) {
 tabObj.items[i].header.iconPosition = e.itemData.text;
 }
 }
</script>

```

### STYLES.CSS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "Twitter", IconCss = "e-
twitter" };
 ViewBag.headerText1 = new TabHeader { Text = "Facebook", IconCss = "e-
facebook" };
 ViewBag.headerText2 = new TabHeader { Text = "WhatsApp", IconCss = "e-
whatsapp" };
 ViewBag.positionData = new string[] { "left", "right", "top", "bottom"
};
 return View();
}

```

Output be like the below.

## Icon Positions

left



TWITTER



FACEBOOK



WHATSAPP

Twitter is an online social networking service that enables users to send and read short 140-character messages called tweets. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app. Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day.

## See Also

- [How to customize selected tab styles](#)

## Orientation

This section explains about modifying the position and modes of Tab header.

It allows placing the header section inside the Tab component at different positions by using the `headerPlacement` property. The available positions are as follows:

- **Top:** Tab header items can be arranged horizontally, and their content can be placed after the header.
- **Bottom:** Tab header items can be arranged horizontally, and their content can be placed before the header.
- **Left:** Tab header items can be arranged vertically, and their content can be placed after the header.
- **Right:** Tab header items can be arranged vertically, and their content can be placed before the header.

It is also adaptable to the available space when the tab items exceed the view space. You can customize the modes by using `overflowMode` property. The available modes are as follows:

- Scrollable
- Popup

## CSHTML

```
@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@using Syncfusion.EJ2.DropDowns;
<div class="col-lg-12 control-section">
 <div id="default" class="e-sample-resize-container">
 <div class='row' style="float:left">
 <label> Header Position </label>
 <div>
```

```

@Html.EJS().DropDownList("headerPosition").Width("90%").DataSource(ViewBag.orientationData).Value("Top").Change("changeHeaderPosition").Render()
 </div>
</div>
<div class='row' style="float:right">
 <label> Mode </label>
</div>

@Html.EJS().DropDownList("Mode").Width("100%").DataSource(ViewBag.positionData).Value("Scrollable").Change("changeOverflowMode").Render()
 </div>
</div>

@Html.EJS().Tab("ej2Tab").HeightAdjustMode(HeightStyles.None).Height("150px").Width("700px").Items(ViewBag.adaptiveItems).Created("tabCreated").Render()
 </div>
</div>
<style>
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }

 #ej2Tab {
 margin-top: 80px;
 margin-left: 80px;
 }

 #default {
 margin-top: 15px;
 }

 #default .row {
 margin-right: 20px;
 margin-left: 20px;
 }
</style>
<script type="text/javascript">
 var tabObj;
 function tabCreated() {
 tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 }
 // Change event function for DropDownList component
 function changeHeaderPosition(e) {
 tabObj.headerPlacement = e.itemData.value;
 tabObj.dataBind();
 }
 function changeOverflowMode(e) {
 tabObj.overflowMode = e.itemData.value;
 tabObj.dataBind();
 }
</script>

```

## ORIENTATION.CS

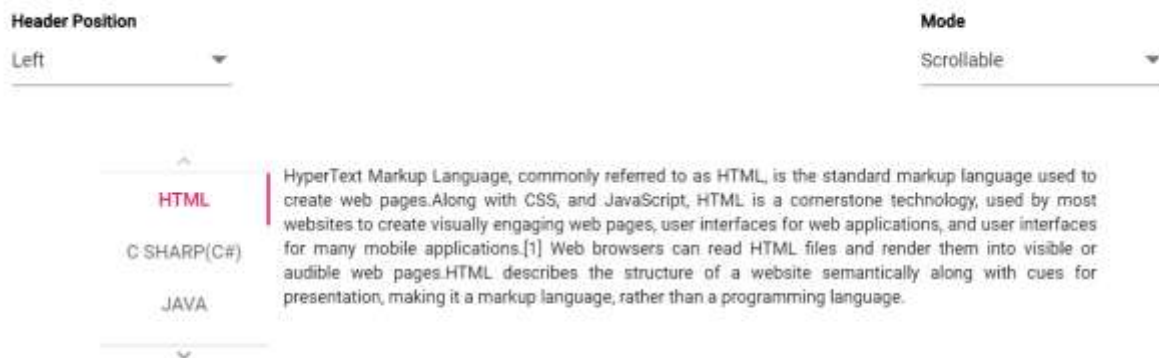
```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Navigations;
// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
namespace EJ2CoreSampleBrowser.Controllers
{
 public partial class TabController : Controller
 {
 List<TabTabItem> adaptiveItems = new List<TabTabItem>();
 public IActionResult ResponsiveModes()
 {
 ViewBag.positionData = new string[] { "Scrollable", "Popup" };
 ViewBag.orientationData = new string[] { "Top",
"Bottom", "Left", "Right" };
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp(C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
 ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
 ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
 ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
 ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "HTML" }, Content = "HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web pages.Along with
CSS, and JavaScript, HTML is a cornerstone technology, used by most websites
to create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages.HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "C Sharp(C#)" }, Content = "C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
is led by Anders Hejlsberg.The most recent version is C# 5.0, which was
released on August 15, 2012." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "Java" }, Content = "Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross - platform computing environment.Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers.While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "VB.Net" }, Content = "The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also includes a command - line
VB.NET compiler.The most recent version is VB 2012, which was released on
August 15, 2012." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "Xamarin" }, Content = "Xamarin is a San Francisco, California based
software company created in May 2011[3] by the engineers that created Mono,

```

```
[4] Mono for Android and MonoTouch that are cross - platform implementations
of the Common Language Infrastructure (CLI) and Common Language
Specifications (often called Microsoft.NET). With a C#-shared codebase,
developers can use Xamarin tools to write native Android, iOS, and Windows
apps with native user interfaces and share code across multiple
platforms. [5] Xamarin has over 1 million developers in more than 120
countries around the World as of May 2015." });
adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "ASP.NET" }, Content = "ASP.NET is an open-source server-side web
application framework designed for web development to produce dynamic web
pages. It was developed by Microsoft to allow programmers to build dynamic
web sites, web applications and web services. It was first released in
January 2002 with version 1.0 of the .NET Framework, and is the successor to
Microsoft Active Server Pages (ASP) technology. ASP.NET is built on the
Common Language Runtime (CLR), allowing programmers to write ASP.NET code
using any supported .NET language. The ASP.NET SOAP extension framework
allows ASP.NET components to process SOAP messages." });
adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "ASP.NET MVC" }, Content = "The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the model-view-controller (MVC)
pattern. It is open - source software, apart from the ASP.NET Web Forms
component which is proprietary. In the later versions of ASP.NET, ASP.NET
MVC, ASP.NET Web API, and ASP.NET Web Pages (a platform using only Razor
pages) will merge into a unified MVC 6. The project is called ASP.NET vNext."
});
adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "JavaScript" }, Content = "JavaScript (JS) is an interpreted computer
programming language. It was originally implemented as part of web browsers
so that client - side scripts could interact with the user, control the
browser, communicate asynchronously, and alter the document content that was
displayed. [5] More recently, however, it has become common in both game
development and the creation of desktop applications." });
ViewBag.adaptiveItems = adaptiveItems;
return View();
}
}
```

Output be like the below.



## Localization

Localization library allows to localize the default text content of Tab. In Tab, The close button's tooltip text alone will be localize based on culture.

| Locale key | en-US (default) |

|-----|-----|

| closeButtonTitle | Close |

## Loading translations

To load translation object in an application use `load` function of `L10n` class.

In the below sample, `French` culture is set to Tab and change the close button's tooltip text.

## CSHTML

```
@using Syncfusion.EJ2.Navigations;
@Html.EJS().Tab("ej2Tab").Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerText0, Content = "HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language." },
 new TabTabItem { Header = ViewBag.headerText1, Content = "C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 5.0, which was released on August 15, 2012." },
 new TabTabItem { Header = ViewBag.headerText2, Content = "Java is a set of computer software and specifications developed by Sun Microsystems, later acquired by Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While less common, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages." }
}).HeightAdjustMode(HeightStyles.Auto).ShowCloseButton(true).Locale("fr-BE").Render()
<script>
 window.onload = function () {
 ej.base.L10n.load({
 'fr-BE': {
 'tab': { 'closeButtonTitle': "Fermer" }
 }
 });
 }
</script>
<style>
 #container {
 visibility: hidden;
 max-width: 650px;
 }
 #loader {
```



```

 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>

```

### LOCALIZATION.CS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp (C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 return View();
}

```

### Drag and drop items

The Tab component allows you to drag and drop any item by setting [allowDragAndDrop](#) to **true**. Items can be reordered to any place by dragging and dropping them onto the desired location.

- If you need to prevent dragging action for a particular item, the [onDragStart](#) event can be used which will trigger when the item drag is started. If you need to prevent dropping action for a particular item, the [dragged](#) event can be used which will trigger when the drag action is stopped.
- The [dragArea](#) defines the area in which the draggable element movement will be occurring. Outside that area will be restricted for the draggable element movement.
- The [onDragStart](#) event will be triggered before dragging the item from Tab.
- The [dragging](#) event will be triggered when the Tab item is being dragged.
- The [dragged](#) event will be triggered when the Tab item is dropped on the target element successfully.

In the following sample, the [allowDragAndDrop](#) property is enabled.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
@{
 var contentOne = "India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India

```

```

Andaman and Nicobar Islands share a maritime border with Thailand and
Indonesia.";
 var contentTwo = "Australia, officially the Commonwealth of Australia,
is a country comprising the mainland of the Australian continent, the island
of Tasmania and numerous smaller islands. It is the world sixth-largest
country by total area. Neighboring countries include Indonesia, East Timor
and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New
Caledonia to the north-east; and New Zealand to the south-east.";
 var contentThree = "The United States of America (USA or U.S.A.),
commonly called the United States (US or U.S.) and America, is a federal
republic consisting of fifty states and a federal district. The 48
contiguous states and the federal district of Washington, D.C. are in
central North America between Canada and Mexico. The state of Alaska is west
of Canada and east of Russia across the Bering Strait, and the state of
Hawaii is in the mid-North Pacific. The country also has five populated and
nine unpopulated territories in the Pacific and the Caribbean.";
 var contentFour = "France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.";
}
<div id='container'>
 @(Html.EJS().Tab("draggableTab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextOne, Content =
@contentOne },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content =
@contentTwo },
 new TabTabItem { Header = ViewBag.headerTextThree, Content =
@contentThree },
 new TabTabItem { Header = ViewBag.headerTextFour, Content =
@contentFour }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .DragArea("#container")
 .AllowDragAndDrop(true)
 .Render()
)
</div>
<style>
 #draggableTab .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>

```

## DRAGANDDROP.CS

```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "India" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Australia" };
}

```

```

ViewBag.headerTextThree = new TabHeader { Text = "USA" };
ViewBag.headerTextFour = new TabHeader { Text = "France" };
return View();
}

```

Output be like the below.



### Drag and drop item between tabs

It is possible to drag and drop the tab items between two tabs, by manually saving those dropped items as new tab item data through the `addTab` method of Tab and removing the dragged item through the `removeTab` method of Tab.

In this example, we have used the tab control as an external source, and the item from the tab component is dragged and dropped onto another Tab. Therefore, it is necessary to use the `onDragStart` and `dragged` event of the Tab component, where we can form an event object and save it using the `addTab` method of the Tab and remove the dragged item through `removeTab` method of Tab using the dragged item index.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
@{
 var contentOne = "India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west;China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.";
 var contentTwo = "Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.";
 var contentThree = "The United States of America (USA or U.S.A.), commonly called the United States (US or U.S.) and America, is a federal republic consisting of fifty states and a federal district. The 48 contiguous states and the federal district of Washington, D.C. are in central North America between Canada and Mexico. The state of Alaska is west of Canada and east of Russia across the Bering Strait, and the state of Hawaii is in the mid-North Pacific. The country also has five populated and nine unpopulated territories in the Pacific and the Caribbean.";
}

```

```

 var contentFour = "France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.";

 var contentFive = "HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web pages. Along with
CSS, and JavaScript, HTML is a cornerstone technology, used by most websites
to create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages. HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language.";

 var contentSix = "C# is intended to be a simple, modern, general-
purpose, object-oriented programming language. Its development team is led
by Anders Hejlsberg. The most recent version is C# 5.0, which was released
on August 15, 2012.";

 var contentSeven = "Java is a set of computer software and
specifications developed by Sun Microsystems, later acquired by Oracle
Corporation, that provides a system for developing application software and
deploying it in a cross-platform computing environment. Java is used in a
wide variety of computing platforms from embedded devices and mobile phones
to enterprise servers and supercomputers. While less common, Java applets
run in secure, sandboxed environments to provide many features of native
applications and can be embedded in HTML pages.";

 var contentEight = "The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also includes a command-line
VB.NET compiler. The most recent version is VB 2012, which was released on
August 15, 2012.";
}
<div id='container'>
 @(Html.EJS().Tab("firstTab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextOne, Content =
@contentOne },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content =
@contentTwo },
 new TabTabItem { Header = ViewBag.headerTextThree, Content =
@contentThree },
 new TabTabItem { Header = ViewBag.headerTextFour, Content =
@contentFour }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .DragArea("#container")
 .AllowDragAndDrop(true)
 .OnDragStart("firstTabdragStart")
 .Dragged("firstTabDragStop")
 .Render()
)
 @(Html.EJS().Tab("secondTab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextFive, Content =
@contentFive },
 new TabTabItem { Header = ViewBag.headerTextSix, Content =
@contentSix },

```

```

 new TabTabItem { Header = ViewBag.headerTextSeven, Content =
@contentSeven },
 new TabTabItem { Header = ViewBag.headerTextEight, Content =
@contentEight }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .DragArea("#container")
 .AllowDragAndDrop(true)
 .OnDragStart("secondTabDragStart")
 .Dragged("secondTabDragStop")
 .Render()
)
</div>
<script>
 var firstTabObj;
 var secondTabObj;
 var firstTabitem;
 var secondTabitem;
 var dragItemIndex;
 var dragItemContainer;
 function firstTabdragStart(args) {
 firstTabObj = document.getElementById("firstTab").ej2_instances[0];
 firstTabitem = [firstTabObj.items[args.index]];
 args.draggedItem.style.visibility = 'hidden';
 dragItemContainer = args.draggedItem.closest('.e-tab');
 }
 function firstTabDragStop(args) {
 if (!ej.base.isNullOrUndefined(args.target.closest('.e-tab')) &&
!dragItemContainer.isSameNode(args.target.closest('.e-tab'))) {
 args.cancel = true;
 var tabElement = args.target.closest('.e-tab');
 var dropItem = args.target.closest('.e-toolbar-item');
 if (tabElement != null && dropItem != null) {
 firstTabObj =
document.getElementById("firstTab").ej2_instances[0];
 secondTabObj =
document.getElementById("secondTab").ej2_instances[0];
 dragItemIndex =
Array.prototype.indexOf.call(firstTabObj.element.querySelectorAll('.e-
toolbar-item'), args.draggedItem);
 var dropItemContainer = args.target.closest('.e-toolbar-
items');
 var dropItemIndex = (dropItemContainer != null) ?
(Array.prototype.slice.call(dropItemContainer.querySelectorAll('.e-toolbar-
item'))).indexOf(dropItem) : '';
 secondTabObj.addTab(firstTabitem, dropItemIndex);
 firstTabObj.removeTab(dragItemIndex);
 }
 }
 }
 function secondTabDragStart(args) {
 secondTabObj =
document.getElementById("secondTab").ej2_instances[0];
 secondTabitem = [secondTabObj.items[args.index]];
 args.draggedItem.style.visibility = 'hidden';
 dragItemContainer = args.draggedItem.closest('.e-tab');
 }
}

```

```

function secondTabDragStop(args) {
 if (!ej.base.isNullOrUndefined(args.target.closest('.e-tab')) &&
 !dragItemContainer.isSameNode(args.target.closest('.e-tab'))) {
 args.cancel = true;
 var tabElement = args.target.closest('.e-tab');
 var dropItem = args.target.closest('.e-toolbar-item');
 if (tabElement != null && dropItem != null) {
 firstTabObj =
document.getElementById("firstTab").ej2_instances[0];
 secondTabObj =
document.getElementById("secondTab").ej2_instances[0];
 dragItemIndex =
Array.prototype.indexOf.call(secondTabObj.element.querySelectorAll('.e-
toolbar-item'), args.draggedItem);
 var dropItemContainer = args.target.closest('.e-toolbar-
items');
 var dropItemIndex = (dropItemContainer != null) ?
(Array.prototype.slice.call(dropItemContainer.querySelectorAll('.e-toolbar-
item')).indexOf(dropItem) : '');
 firstTabObj.addTab(secondTabitem, dropItemIndex);
 secondTabObj.removeTab(dragItemIndex);
 }
 }
}
</script>
<style>
#firstTab .e-content .e-item,
#secondTab .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
}
#firstTab {
 margin-bottom: 40px;
}
#secondTab {
 margin-top: 40px;
}
</style>

```

### TABTOTAB.CS

```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "India" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Australia" };
 ViewBag.headerTextThree = new TabHeader { Text = "USA" };
 ViewBag.headerTextFour = new TabHeader { Text = "France" };
 ViewBag.headerTextFive = new TabHeader { Text = "HTML" };
 ViewBag.headerTextSix = new TabHeader { Text = "C Sharp(C#)" };
 ViewBag.headerTextSeven = new TabHeader { Text = "Java" };
 ViewBag.headerTextEight = new TabHeader { Text = "VB.Net" };
 return View();
}

```

Output be like the below.



### Drag and drop items to external source

It is possible to drag and drop the items to any of the external sources from the Tab, by manually saving those dropped items as new node data through the `addNodes` method of Treeview and removing the dragged item through the `removeTab` method of Tab.

In this example, we have used the tree view control as an external source, and the item from the tab component is dragged and dropped onto the child nodes of the tree view component. Therefore, it is necessary to use the `dragged` event of the Tab component, where we can form an event object and save it using the `addNodes` method of the Treeview and remove the dragged item through the `removeTab` method of Tab using the dragged item index.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@{
 var contentOne = "India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west;China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.";
 var contentTwo = "Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New Caledonia to the north-east; and New Zealand to the south-east.";
 var contentThree = "The United States of America (USA or U.S.A.), commonly called the United States (US or U.S.) and America, is a federal republic consisting of fifty states and a federal district. The 48 contiguous states and the federal district of Washington, D.C. are in central North America between Canada and Mexico. The state of Alaska is west of Canada and east of Russia across the Bering Strait, and the state of
```

```

Hawaii is in the mid-North Pacific. The country also has five populated and
nine unpopulated territories in the Pacific and the Caribbean.";
 var contentFour = "France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.";
}
<div id='container'>
 @(Html.EJS().Tab("draggableTab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextOne, Content =
@contentOne },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content =
@contentTwo },
 new TabTabItem { Header = ViewBag.headerTextThree, Content =
@contentThree },
 new TabTabItem { Header = ViewBag.headerTextFour, Content =
@contentFour }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .DragArea("#container")
 .AllowDragAndDrop(true)
 .Created("onTabCreate")
 .Dragged("tabDragStop")
 .Render()
)
 @(Html.EJS().TreeView("draggableTreeview").CssClass("treeview-external-
drop-tab").Fields(field =>
field.DataSource(ViewBag.data).Id("id").Text("text")).Render()
</div>
<script>
 var i = 0;
 function onTabCreate() {
 var tabElement = document.getElementById('draggableTab');
 if (!ej.base.isNullOrUndefined(tabElement)) {
 tabElement.querySelector('.e-tab-header').classList.add('e-
draggable');
 tabElement.querySelector('.e-content').classList.add('tab-
content');
 }
 }
 function tabDragStop(args) {
 var tabObj =
document.getElementById("draggableTab").ej2_instances[0];
 var treeObj =
document.getElementById("draggableTreeview").ej2_instances[0];
 var dragTabIndex =
Array.prototype.indexOf.call(tabObj.element.querySelectorAll('.e-toolbar-
item'), args.draggedItem);
 var dragItem = tabObj.items[dragTabIndex];
 var dropNode = args.target.closest('#draggableTreeview .e-list-
item');
 if (dropNode != null && !args.target.closest('#draggableTab .e-
toolbar-item')) {

```



```

 args.cancel = true;
 var dropContainer = (document.querySelector('.treeview-external-
drop-tab')).querySelectorAll('.e-list-item');
 var dropIndex = Array.prototype.indexOf.call(dropContainer,
dropNode);
 var newNode = [{ id: 'list' + i, text: dragItem.header.text }];
 tabObj.removeTab(dragTabIndex);
 treeObj.addNodes(newNode, 'Treeview', dropIndex);
 }
}
</script>
<style>
#draggableTab .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
}
.treeview-external-drop-tab .e-list-item,
.e-bigger .treeview-external-drop-tab .e-list-item {
 border: 0.5px solid #E1E7EC;
 line-height: 15px;
 padding: 0 5px;
}
.treeview-external-drop-tab .e-list-item.e-hover > .e-fullrow,
.treeview-external-drop-tab .e-list-item.e-active > .e-fullrow,
.treeview-external-drop-tab .e-list-item.e-active.e-hover > .e-
fullrow,
.e-bigger .treeview-external-drop-tab .e-list-item.e-hover > .e-
fullrow,
.e-bigger .treeview-external-drop-tab .e-list-item.e-active > .e-
fullrow,
.e-bigger .treeview-external-drop-tab .e-list-item.e-active.e-hover
> .e-fullrow {
 background-color: transparent;
 border-color: transparent;
 box-shadow: none !important;
}
#draggableTab {
 margin-bottom: 40px;
}
#draggableTreeview {
 margin-top: 40px;
}
</style>

```

### TABTOTREEVIEW.CS

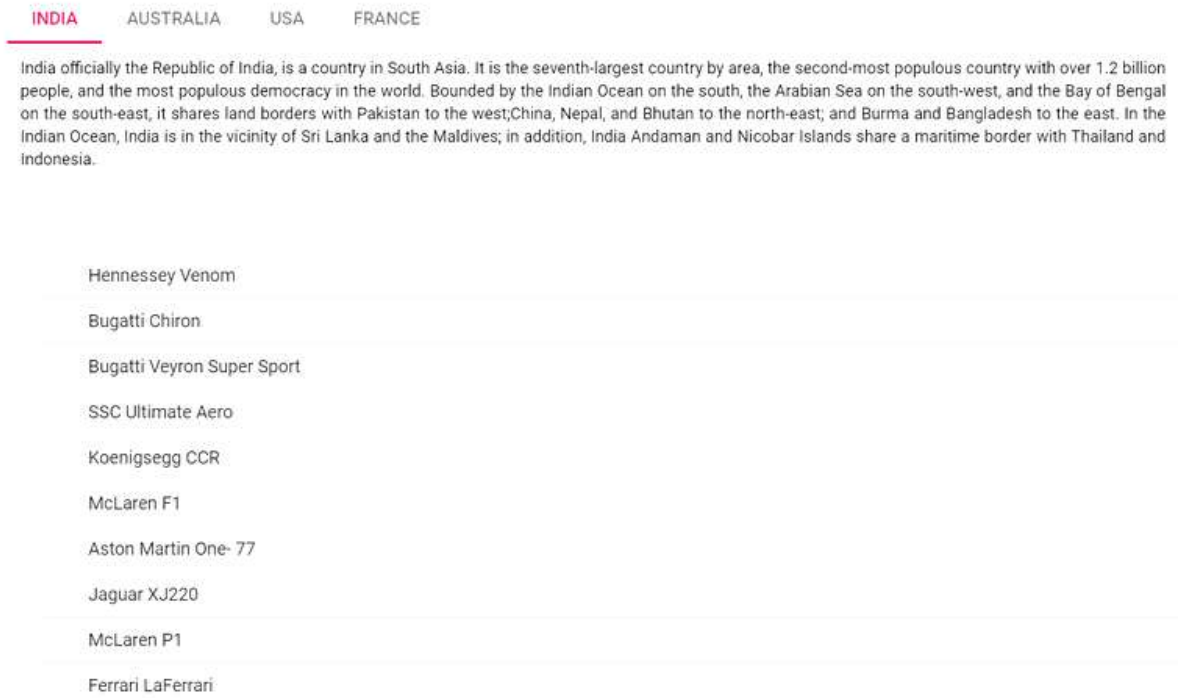
```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "India" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Australia" };
 ViewBag.headerTextThree = new TabHeader { Text = "USA" };
 ViewBag.headerTextFour = new TabHeader { Text = "France" };
 List<object> componentList = new List<object>();
 componentList.Add(new
 {

```

```
 text = "Hennessey Venom",
 id = "list-01"
 });
 componentList.Add(new
 {
 text = "Bugatti Chiron",
 id = "list-02"
 });
 componentList.Add(new
 {
 text = "Bugatti Veyron Super Sport",
 id = "list-03"
 });
 componentList.Add(new
 {
 text = "SSC Ultimate Aero",
 id = "list-04"
 });
 componentList.Add(new
 {
 text = "Koenigsegg CCR",
 id = "list-05"
 });
 componentList.Add(new
 {
 text = "McLaren F1",
 id = "list-06"
 });
 componentList.Add(new
 {
 text = "Aston Martin One- 77",
 id = "list-07"
 });
 componentList.Add(new
 {
 text = "Jaguar XJ220",
 id = "list-08"
 });
 componentList.Add(new
 {
 text = "McLaren P1",
 id = "list-09"
 });
 componentList.Add(new
 {
 text = "Ferrari LaFerrari",
 id = "list-10"
 });
 ViewBag.data = componentList;
 return View();
}
```

Output be like the below.



### Drag and drop items from external source

It is possible to drag and drop the items from any of the external sources into the Tab, by manually saving those dropped items as new item data through the `addTab` method of Tab and removing the dragged node through the `removeNodes` method of Treeview.

In this example, we have used the tree view control as an external source, and the child nodes from the tree view component are dragged and dropped onto the Tab. Therefore, it is necessary to use the `nodeDragStop` event of the Treeview component, where we can form an event object and save it using the `addTab` method of the Tab and remove the dragged node through the `removeNodes` method of Treeview.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@{
 var contentOne = "India officially the Republic of India, is a country in South Asia. It is the seventh-largest country by area, the second-most populous country with over 1.2 billion people, and the most populous democracy in the world. Bounded by the Indian Ocean on the south, the Arabian Sea on the south-west, and the Bay of Bengal on the south-east, it shares land borders with Pakistan to the west; China, Nepal, and Bhutan to the north-east; and Burma and Bangladesh to the east. In the Indian Ocean, India is in the vicinity of Sri Lanka and the Maldives; in addition, India Andaman and Nicobar Islands share a maritime border with Thailand and Indonesia.";
 var contentTwo = "Australia, officially the Commonwealth of Australia, is a country comprising the mainland of the Australian continent, the island of Tasmania and numerous smaller islands. It is the world sixth-largest country by total area. Neighboring countries include Indonesia, East Timor
```

```

and Papua New Guinea to the north; the Solomon Islands, Vanuatu and New
Caledonia to the north-east; and New Zealand to the south-east.";
 var contentThree = "The United States of America (USA or U.S.A.),
commonly called the United States (US or U.S.) and America, is a federal
republic consisting of fifty states and a federal district. The 48
contiguous states and the federal district of Washington, D.C. are in
central North America between Canada and Mexico. The state of Alaska is west
of Canada and east of Russia across the Bering Strait, and the state of
Hawaii is in the mid-North Pacific. The country also has five populated and
nine unpopulated territories in the Pacific and the Caribbean.";
 var contentFour = "France, officially the French Republic is a sovereign
state comprising territory in western Europe and several overseas regions
and territories. The European part of France, called Metropolitan France,
extends from the Mediterranean Sea to the English Channel and the North Sea,
and from the Rhine to the Atlantic Ocean; France covers 640,679 square kilo
metres and as of August 2015 has a population of 67 million, counting all
the overseas departments and territories.";
}
<div id='container'>
 @(Html.EJS().Tab("draggableTab")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerTextOne, Content =
@contentOne },
 new TabBarItem { Header = ViewBag.headerTextTwo, Content =
@contentTwo },
 new TabBarItem { Header = ViewBag.headerTextThree, Content =
@contentThree },
 new TabBarItem { Header = ViewBag.headerTextFour, Content =
@contentFour }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .DragArea("#container")
 .Render()
)

@Html.EJS().TreeView("draggableTreeview").AllowDragAndDrop(true).DragArea("#
container").NodeDragStop("onNodeDragStop").NodeDragging("onNodeDrag").CssCla
ss("treeview-external-drop-tab").Fields(field =>
field.DataSource(ViewBag.data).Id("id").Text("text")).Render()
</div>
<script>
 function onNodeDragStop(args) {
 var tabObj =
document.getElementById("draggableTab").ej2_instances[0];
 var treeObj =
document.getElementById("draggableTreeview").ej2_instances[0];
 var dropElement = args.target.closest('#draggableTab .e-toolbar-
item');
 if (dropElement != null) {
 var tabElement = document.querySelector('#draggableTab');
 var dropItemIndex =
[[]].slice.call(tabElement.querySelectorAll('.e-toolbar-
item')).indexOf(dropElement);
 var newTabItem = [{
 header: { 'text': args.draggedNodeData.text.toString() },
 content: args.draggedNodeData.text.toString() + ' Content'
 }];

```

```

 tabObj.addTab(newTabItem, dropItemIndex);
 treeObj.removeNodes([args.draggedNode]);
 args.cancel = true;
 } else {
 var dropNode = args.target.closest('#draggableTreeview .e-list-
item ');
 if (!ej.base.isNullOrUndefined(dropNode) && args.dropIndicator
=== 'e-drop-in') {
 args.cancel = true;
 }
 }
}
function onNodeDrag(args) {
 if (!ej.base.isNullOrUndefined(args.target.closest('.tab-content')))
{
 args.dropIndicator = 'e-no-drop';
 } else if
(!ej.base.isNullOrUndefined(args.target.closest('#draggableTab .e-tab-
header')))) {
 args.dropIndicator = 'e-drop-in';
 }
}
</script>
<style>
#draggableTab .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
}
.treeview-external-drop-tab .e-list-item,
.e-bigger .treeview-external-drop-tab .e-list-item {
 border: 0.5px solid #E1E7EC;
 line-height: 15px;
 padding: 0 5px;
}
.treeview-external-drop-tab .e-list-item.e-hover > .e-fullrow,
.treeview-external-drop-tab .e-list-item.e-active > .e-fullrow,
.treeview-external-drop-tab .e-list-item.e-active.e-hover > .e-
fullrow,
.e-bigger .treeview-external-drop-tab .e-list-item.e-hover > .e-
fullrow,
.e-bigger .treeview-external-drop-tab .e-list-item.e-active > .e-
fullrow,
.e-bigger .treeview-external-drop-tab .e-list-item.e-active.e-hover
> .e-fullrow {
 background-color: transparent;
 border-color: transparent;
 box-shadow: none !important;
}
#draggableTab {
 margin-bottom: 40px;
}
#draggableTreeview {
 margin-top: 40px;
}
</style>

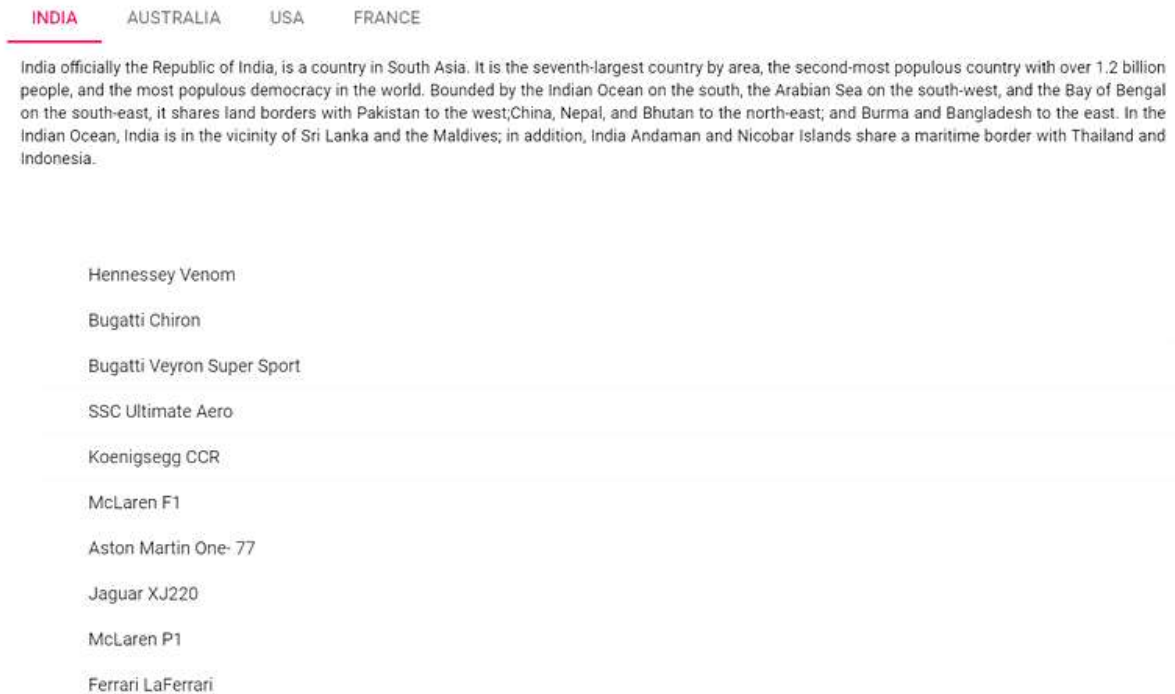
```

**TREEVIEWTOTAB.CS**

```
public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "India" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Australia" };
 ViewBag.headerTextThree = new TabHeader { Text = "USA" };
 ViewBag.headerTextFour = new TabHeader { Text = "France" };
 List<object> componentList = new List<object>();
 componentList.Add(new
 {
 text = "Hennessey Venom",
 id = "list-01"
 });
 componentList.Add(new
 {
 text = "Bugatti Chiron",
 id = "list-02"
 });
 componentList.Add(new
 {
 text = "Bugatti Veyron Super Sport",
 id = "list-03"
 });
 componentList.Add(new
 {
 text = "SSC Ultimate Aero",
 id = "list-04"
 });
 componentList.Add(new
 {
 text = "Koenigsegg CCR",
 id = "list-05"
 });
 componentList.Add(new
 {
 text = "McLaren F1",
 id = "list-06"
 });
 componentList.Add(new
 {
 text = "Aston Martin One- 77",
 id = "list-07"
 });
 componentList.Add(new
 {
 text = "Jaguar XJ220",
 id = "list-08"
 });
 componentList.Add(new
 {
 text = "McLaren P1",
 id = "list-09"
 });
 componentList.Add(new
```

```
{
 text = "Ferrari LaFerrari",
 id = "list-10"
});
ViewBag.data = componentList;
return View();
}
```

Output be like the below.



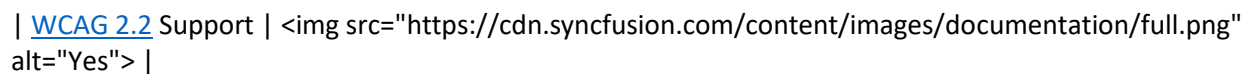
## Accessibility ASP.NET MVC Tab control

The Tab control followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Tab control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes" |

| [Section 508](#) Support |  alt="Yes" |

| Screen Reader Support |  alt="Yes" |

| Right-To-Left Support |  alt="Yes" |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the control meet the requirement.</div>

<div> - Some features of the control do not meet the requirement.</div>

<div> - The control does not meet the requirement.</div>

### ARIA attributes

Tab control is designed by considering [WAI-ARIA](#) standard. Tab is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices.

The following list of attributes are added in the Tab.

#### | Roles and Attributes | Functionalities |

| --- | --- |

| tablist | Attribute is set to the Tab header element that describes actual role of the element. |

| tab | Attribute is set to the Tab items element to indicates an interactive element inside a **tablist** that, when activated, displays its associated **tabpanel**. |

| tabpanel | Attribute is set to the Tab content that describes the role for viewing the active content. |

| aria-orientation | Attribute is set to the Tab header element indicates the Tab header orientation. Default value of this attribute is horizontal. |

| aria-selected | Attribute set to the Tab items to indicates the selection state for Tab items. Active Tab is set to true for this attribute. |



| aria-labelledby | Attribute is set to the Tab content element to indicates the associated Tab header for the content. |

| aria-controls | Attribute is set to the Tab items element to indicates the associated **tabpanel** for the header. |

| aria-haspopup | Attribute is set to the Popup element to indicates the popup mode in the Tab. The default value of this attribute is false. If popup mode is enabled, the attribute value is set to true. |

| aria-disabled | Attribute set to the Tab items to It indicates the disabled state of the Tab. |

### Keyboard interaction

By default, keyboard navigation is enabled. This control implements keyboard navigation support by following the WAI-ARIA practices. Once focused on the active Tab element, you can use the following key combination for interacting with the Tab.

| Key   | Description |
|-------|-------------|
| ----- | -----       |

| Left | Moves focus to the previous Tab. If focus is on the first Tab, the focus will not move to any Tab. |

| Right | Moves focus to the next Tab. If focus is on the last Tab element, the focus will not move to any Tab. |

| Enter or Space | Selects the Tab if it is not selected. Opens the popup dropdown icon if it is focussed. Select the Tab item as active when popup item is focussed. |

| Esc(Escape) | Closes the popup if popup is in opened state. |

| Down or Up | When the popup is open and focused, it will move to previous/next Tab items of the popup in the vertical direction. |

| Home | Moves focus to the first Tab. |

| End | Moves focus to the last Tab. |

| Shift + F10 | If popup mode is enabled, it opens the popup when the Tab is focused. |

| Delete | Deletes the Tab, if close button is enabled in Tab header. |

| Tab | To Move focus through the interactive elements. |

| Shift + Tab | To Move focus through the interactive elements. |

### Ensuring accessibility

The Tab control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tab control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tab control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

### CSS Structure

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

#### Customizing Tab

Use the following CSS to customize the Tab.

`CSS

```
.e-tab {
border: 5px solid rgb(173, 255, 47);
}
```

,

#### Customizing the Tab items

Use the following CSS to customize the header items of Tab.

`CSS

```
.e-tab .e-tab-header .e-toolbar-items {
background: #9faed8;
border: 2px solid blue;
}
```

,

Use the following CSS to customize the content items of Tab.

`CSS

```
.e-tab .e-content .e-item {
color: #a78515;
font-size: 14px;
}
```

,

#### Customizing Tab's header

Use the following CSS to customize the header of Tab control.

`CSS

```
.e-tab .e-tab-header {
background: #badfba !important;
}
```

,

#### Customizing Tab's header icon

Use the following CSS to customize the header item icon of Tab control.

`CSS

```
.e-tab .e-tab-header .e-toolbar-item .e-tab-icon {
color: #badfba !important;
}
`
```

### Customizing Tab's content

Use the following CSS to customize the content of Tab control.

```
`CSS
.e-tab .e-content {
background: #d1f6d1 !important;
}
`
```

### Customizing the hover state of Tab control

Use the following CSS to customize the tab item when hovering.

```
`CSS
.e-tab .e-tab-header .e-toolbar-item .e-tab-wrap:hover {
background: #d1f6d1 !important;
}
`
```

Use the following CSS to customize the tab item popup icon when hovering.

```
`CSS
.e-tab .e-tab-header .e-hor-nav .e-popup-up-icon:hover,
.e-tab .e-tab-header .e-hor-nav .e-popup-down-icon:hover {
background: #d1f6d1 !important;
}
`
```

### Customizing selected item of Tab control

Use the following CSS to customize the selected tab item.

```
`CSS
.e-tab .e-tab-header .e-toolbar-item.e-active {
background: #d1f4d1;
}
`
```

Use the following CSS to customize the selected tab item text and icon.

```
`CSS
```

```
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-text,
.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-icon {
color: green !important;
}
`
```

## How To

### Load content through Ajax

The Tab supports to load external contents through AJAX library. Refer to the following steps.

- Import the Ajax module from ej2-base and initialize with URL path.
- Get the data from Ajax Success event, then initialize the Tab with retrieved external path data.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@Html.EJS().Tab("MainTab").Created("Created").Render()
<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>
<script>
 function Created() {
 var TabObj = document.getElementById("MainTab").ej2_instances[0];
 var ajax = new ej.base.Ajax('./network.html', 'GET', true);
 ajax.send().then();
 ajax.onSuccess = function (data) {
 TabObj.addTab([{ header: { 'text': 'Twitter' }, content: data }],
0);
 }
 }
</script>
```

### AJAX.CS

```
public IActionResult Index()
{
 return View();
}
```

```
}

```

## Prevent content swipe selection

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().Tab("ej2Tab")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerTextOne, Content = "Twitter
is an online social networking service that enables users to send and read
short 140-character messages called tweets. Registered users can read and
post tweets, but those who are unregistered can only read them. Users access
Twitter through the website interface, SMS or mobile device app Twitter Inc.
is based in San Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone,
and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, with more than 100 million users posting 340 million
tweets a day in 2012.The service also handled 1.6 billion search queries per
day." },
 new TabBarItem { Header = ViewBag.headerTextTwo, Content = "Facebook
is an online social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College roommates and fellow students Eduardo Saverin,
Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders had
initially limited the website's membership to Harvard students, but later
expanded it to colleges in the Boston area, the Ivy League, and Stanford
University. It gradually added support for students at various other
universities and later to high-school students." },
 new TabBarItem { Header = ViewBag.headerTextThree, Content =
"WhatsApp Messenger is a proprietary cross-platform instant messaging client
for smartphones that operates under a subscription business model. It uses
the Internet to send text messages, images, video, user location and audio
media messages to other users using standard cellular mobile numbers. As of
February 2016, WhatsApp had a user base of up to one billion,[10] making it
the most globally popular messaging application. WhatsApp Inc., based in
Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion." }
 })
 .Selecting("tabSelecting")
 .Render()
)
<style>
 .e-content .e-item {
 font-size: 12px;
 padding: 10px;
 text-align: justify;
 }
 .container {
 min-width: 350px;
 max-width: 500px;
 margin: 0 auto;
 }
</style>
<script type="text/javascript">
 function tabSelecting(e) {
 if (e.isSwiped) {
```

```

 e.cancel = true;
 }
}
</script>

```

## SWIPE.CS

```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "Twitter" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Facebook" };
 ViewBag.headerTextThree = new TabHeader { Text = "Whatsapp" };
 return View();
}

```

### Customize selected tab styles

You can customize the Tab style by overriding its header and active tab CSS classes. Define HTML string for adding animation and customizing the Tab header and pass it to [text](#) property. Now you can override the style using custom CSS classes added to the Tab elements.

## CSHTML

```

@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Tab("ej2Tab")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerTextOne, Content = "Andrew received his BTS commercial in 1974 and a Ph.D. in international marketing from the University of Dallas in 1981.He is fluent in French and Italian and reads German.He joined the company as a sales representative, was promoted to sales manager in January 1992 and to vice president of sales in March 1993.Andrew is a member of the Sales Management Roundtable, the Seattle Chamber of Commerce, and the Pacific Rim Importers Association." },
 new TabBarItem { Header = ViewBag.headerTextTwo, Content = "Margaret holds a BA in English literature from Concordia College (1958) and an MA from the American Institute of Culinary Arts (1966).She was assigned to the London office temporarily from July through November 1992." },
 new TabBarItem { Header = ViewBag.headerTextThree, Content = "Janet has a BS degree in chemistry from Boston College (1984).She has also completed a certificate program in food retailing management.Janet was hired as a sales associate in 1991 and promoted to sales representative in February 1992." },
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .OverflowMode(OverflowMode.Popup)
 .Render()
)
<style type="text/css">
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;

```

```

 top: 45%;
 width: 30%;
 }
 #template-wrap {
 display: flex;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
 .container {
 min-width: 350px;
 max-width: 500px;
 margin: 0 auto;
 }
 .e-image {
 background-size: 33px;
 width: 33px;
 height: 33px;
 margin: 0 auto;
 }
 .e-image.e-andrew {
 /* csslint allow: adjoining-classes */
 background-image:
url('https://ej2.syncfusion.com/demos/src/images/employees/3.png');
 }
 .e-image.e-margaret {
 /* csslint allow: adjoining-classes */
 background-image:
url('https://ej2.syncfusion.com/demos/src/images/employees/6.png');
 }
 .e-image.e-janet {
 /* csslint allow: adjoining-classes */
 background-image:
url('https://ej2.syncfusion.com/demos/src/images/employees/7.png');
 }
 #ej2Tab.e-tab .e-toolbar-item .e-title {
 margin-top: 8px;
 }
 #ej2Tab.e-toolbar .e-toolbar-items .e-toolbar-item:not(.e-separator),
 #ej2Tab.e-toolbar .e-toolbar-items .e-toolbar-item .e-tab-wrap {
 width: 125px;
 height: 50px;
 }
 #ej2Tab.e-tab .e-tab-header .e-toolbar-item.e-active .e-tab-wrap {
 /* csslint allow: adjoining-classes */
 background-color: #08c;
 }
 #ej2Tab.e-tab .e-tab-header {
 background-color: #e6e6e6;
 }
 #ej2Tab.e-tab .e-tab-header .e-toolbar-item:not(.e-active) .e-tab-wrap:hover
 {
 background-color: #f2f2f2;
 color: #000;
 }

```

```
#ej2Tab.e-tab .e-toolbar .e-toolbar-items .e-toolbar-item .e-text-wrap {
 height: 50px;
}
#ej2Tab.e-tab .e-toolbar-item.e-active .e-title {
 /* csslint allow: adjoining-classes */
 display: block;
 color: white;
}
#ej2Tab.e-tab .e-toolbar-item .e-text-wrap,
#ej2Tab.e-tab .e-toolbar-item .e-tab-text {
 width: inherit;
 text-align: center;
}
#ej2Tab.e-tab .e-toolbar-item.e-active .e-title.fade-in {
 opacity: 1;
 animation-name: fadeInOpacity;
 animation-iteration-count: 1;
 animation-timing-function: ease-in;
 animation-duration: 0.5s;
}
@@keyframes fadeInOpacity {
 0% {
 opacity: 0;
 }
 100% {
 opacity: 1;
 }
}
</style>
```

### CUSTOMIZE.CS

```
public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "<div id=\"template-
wrap\"><div class='e-image e-andrew'></div><div class='e-title fade-
in'>Andrew</div></div>" };
 ViewBag.headerTextTwo = new TabHeader { Text = "<div id=\"template-
wrap\"><div class='e-image e-margaret'></div><div class='e-title fade-
in'>Margaret</div></div>" };
 ViewBag.headerTextThree = new TabHeader { Text = "<div id=\"template-
wrap\"><div class='e-image e-janet'></div><div class='e-title fade-
in'>Janet</div></div>" };
 return View();
}
```

Output be like the below.





### Create wizard using Tab

In the below Wizard sample, each Tab is integrated with required components to complete the reservation. Each field is provided with validation for all mandatory option to proceed to next tabs. Using Tab item's template property the components are added into content.

Create the following contents for each tab in the wizard.

#### 1. Search tab:

Created with [DropDownList] to select the source, destination and type of ticket. A [DatePicker] for choosing the date of journey.

#### 2. Train tab:

Based on the selected start and end point, populated Grid with random list of available seats and train list. Initially define the columns and row selected event for validating, after the source and destination chosen update the [dataSource] for the Grid.

#### 3. Passenger tab:

A table with Textbox, Numeric, DropDownList for adding passenger name, age, gender and preferred berth/seat. Add validation on entering passenger details to proceed.

#### 4. Payment tab:

Calculate the ticket cost based on location, passenger count and ticket type. Generate data for Grid with passenger details, train number and ticket cost summary.

You can go back on each tab using buttons available in it and tabs are disabled to navigate through tab header click actions. Once you end the wizard all the data get cleared and wizard goes back to starting tab.

### CSHTML

```
@using Syncfusion.EJ2.DropDowns
@using Syncfusion.EJ2.Navigations
<div class="e-tab-section">
 <div class="e-sample-resize-container">
 <div id="booking" style="display: none">
 <div class="wizard-title">Plan your journey</div>
 <div class="responsive-align">
 <div class="row">
```

```

<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
@Html.EJS().DropDownList("startPoint").Placeholder("From").Width("100%").Dat
aSource(
 ViewBag.citiesData).Fields(new
DropDownListFieldSettings { Text = "Name", Value = "Name" }).Render()
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
@Html.EJS().DropDownList("endPoint").Placeholder("To").Width("100%").DataSou
rce(
 ViewBag.citiesData).Fields(new
DropDownListFieldSettings { Text = "Name", Value = "Name" }).Render()
</div>
</div>
<div class="row">
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
 @Html.EJS().DatePicker("date").Placeholder("Journey
Date").Width("100%").Min("ViewBag.min").Max("ViewBag.max").Focus("showDate")
.Render()
</div>
<div class="col-xs-6 col-sm-6 col-lg-6 col-md-6 search-
item">
@Html.EJS().DropDownList("quota").Placeholder("Ticket type").DataSource(
 ViewBag.quota).Fields(new
DropDownListFieldSettings { Text = "Text", Value = "Text" }).Render()
</div>
</div>
<div class="btn-container">
<button id="searchNext" class="e-btn">Search
Train</button>
</div>

</div>
</div>
<div id="selectTrain" style="display: none">
<div class="wizard-title">Select the train from the list </div>
@ (Html.EJS().Grid("availableTrain")
 .Columns(c =>
 {
 c.Field("TrainNo").HeaderText("Train
No").Width("120").Type("number").Add();
 c.Field("Name").HeaderText("Name").Width("140").Add();

c.Field("Departure").HeaderText("Departure").Width("120").Add();

c.Field("Arrival").HeaderText("Arrival").Width("140").Add();

c.Field("Availability").HeaderText("Availability").Width("140").Type("number
").Add();

 })
 .Width("100%")
 .RowSelected("trainSelected")

```

```

 .Render()
)

 <div class="btn-container">
 <button id="goToSearch" class="e-btn">Back</button>
 <button id="bookTickets" class="e-btn">Continue</button>
 </div>

</div>
<div id="details" style="display: none">
 <div class="details-page wizard-title">Enter the passenger
details</div>
 <div id="PassengersList">
 <table id="passenger-table">
 <colgroup>
 <col />
 <col />
 <col />
 <col />
 <col />
 <col />
 </colgroup>
 <thead>
 <tr>
 <th class="name-header">Name</th>
 <th class="age-header">Age</th>
 <th class="gender-header">Gender</th>
 <th class="type-header">Berth Preference</th>
 </tr>
 </thead>
 <tbody>
 <tr>
 <td>
 <input id="pass_name1" class="e-input"
type="text" placeholder="Passenger Name">
 </td>
 <td>
 @Html.EJS().NumericTextBox("pass_age1").Format("n0").Value(18).Min(1).Max(100).ShowSpinButton(false).Render()
 </td>
 <td>
 @Html.EJS().DropDownList("pass_gender1").Text("Male").DataSource(ViewBag.gen
der).Fields(new DropDownListFieldSettings { Text = "Text", Value = "Text"
}).Render()
 </td>
 <td>
 @Html.EJS().DropDownList("pass_berth1").Placeholder("Optional").DataSource(V
iewBag.berth).Fields(new DropDownListFieldSettings { Text = "Text", Value =
"Text" }).Render()
 </td>
 </tr>
 <tr>
 <td>

```

```

<input id="pass_name2" class="e-input"
type="text" placeholder="Passenger Name">
</td>
<td>

@Html.EJS().NumericTextBox("pass_age2").Format("n0").Value(18).Min(1).Max(100).ShowSpinButton(false).Render()

</td>
<td>

@Html.EJS().DropDownList("pass_gender2").Text("Male").DataSource(ViewBag.gender).Fields(new DropDownListFieldSettings { Text = "Text", Value = "Text"
}).Render()

</td>
<td>

@Html.EJS().DropDownList("pass_berth2").Placeholder("Optional").DataSource(ViewBag.berth).Fields(new DropDownListFieldSettings { Text = "Text", Value =
"Text" }).Render()

</td>
</tr>
<tr>
<td>
<input id="pass_name3" class="e-input"
type="text" placeholder="Passenger Name">
</td>
<td>

@Html.EJS().NumericTextBox("pass_age3").Format("n0").Value(18).Min(1).Max(100).ShowSpinButton(false).Render()

</td>
<td>

@Html.EJS().DropDownList("pass_gender3").Text("Male").DataSource(ViewBag.gender).Fields(new DropDownListFieldSettings { Text = "Text", Value = "Text"
}).Render()

</td>
<td>

@Html.EJS().DropDownList("pass_berth3").Placeholder("Optional").DataSource(ViewBag.berth).Fields(new DropDownListFieldSettings { Text = "Text", Value =
"Text" }).Render()

</td>
</tr>
</tbody>
</table>
</div>

<div class="btn-container">
<button id="goBackToBook" class="e-btn">Back</button>
<button id="confirmTickets" class="e-btn">Continue</button>
</div>

</div>
<div id="confirm" style="display: none">
<div class="tab-title1 wizardtitle">Confirm the details and
proceed</div>

```

```

 @ (Html.EJS().Grid("ticketDetailGrid")
 .Width("100%")
 .Columns(c =>
 {
 c.Field("TrainNo").HeaderText("Train
No").Width("120").Type("number").Add();

c.Field("PassName").HeaderText("Name").Width("120").Add();

c.Field("Gender").HeaderText("Gender").Width("120").Add();
 c.Field("Berth").HeaderText("Berth").Width("120").Add();
 })
 .Render()
)

 <div id="amount"></div>

 <div class="btn-container">
 <button id="goBackDetails" class="e-btn">Back</button>
 <button id="makePayment" class="e-btn">Pay</button>
 </div>
</div>
 @ (Html.EJS().Tab("ej2Tab")
 .HeightAdjustMode(HeightStyles.None)
 .Height("390")
 .Created("tabCreated")
 .Selecting("tabSelecting")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextOne, Content =
ViewBag.content1 },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content =
ViewBag.content2 },
 new TabTabItem { Header = ViewBag.headerTextThree, Content =
ViewBag.content3 },
 new TabTabItem { Header = ViewBag.headerTextFour, Content =
ViewBag.content4 }
 })
 .Render()
)
 <div>

 @Html.EJS().Dialog("alertDialog").Header("Success").Target("#ej2Tab").ShowCl
oseIcon(true).Width("250").IsModal(true).Created("dlgCreated").Visible(false
).Render()

 </div>
 </div>
</div>
<script>
 var tabObj;
 var endPoint;
 var alertDlg;
 var locations;
 var ticketType;
 var startPoint;
 var journeyDate;
 var selectedTrain;
 var availTrainGrid;

```

```

var ticketDetailGrid;
var cities = [
 { name: "Chicago", fare: 300 },
 { name: "San Francisco", fare: 125 },
 { name: "Los Angeles", fare: 175 },
 { name: "Seattle", fare: 250 },
 { name: "Florida", fare: 150 }
];
function tabCreated() {
 document.getElementById("searchNext").onclick = function (e) {
 tabNavigations(e);
 };
 document.getElementById("bookTickets").onclick = function (e) {
 tabNavigations(e);
 };
 document.getElementById("confirmTickets").onclick = function (e) {
 tabNavigations(e);
 };
 document.getElementById("makePayment").onclick = function (e) {
 tabNavigations(e);
 };
 document.getElementById("goToSearch").onclick = function (e) {
 tabNavigations(e);
 };
 document.getElementById("goBackToBook").onclick = function (e) {
 tabNavigations(e);
 };
 document.getElementById("goBackDetails").onclick = function (e) {
 tabNavigations(e);
 };
 tabObj = document.getElementById("ej2Tab").ej2_instances[0];
 startPoint = document.getElementById("startPoint").ej2_instances[0];
 endPoint = document.getElementById("endPoint").ej2_instances[0];
 ticketType = document.getElementById("quota").ej2_instances[0];
 journeyDate = document.getElementById("date").ej2_instances[0];
 availTrainGrid =
document.getElementById("availableTrain").ej2_instances[0];
 ticketDetailGrid =
document.getElementById("ticketDetailGrid").ej2_instances[0];
}
function showDate() {
 journeyDate = document.getElementById("date").ej2_instances[0];
 journeyDate.show();
}
function tabSelecting(e) {
 if (e.isSwiped) {
 e.cancel = true;
 }
}
function dlgCreated() {
 alertDlg = document.getElementById("alertDialog").ej2_instances[0];
 alertDlg.content = "Your payment successflly processed";
 alertDlg.buttons = [{
 buttonModel: { content: "Ok", isPrimary: true },
 click: function () {
 alertDlg.hide();
 tabObj.enableTab(0, true);
 tabObj.enableTab(1, false);
 tabObj.enableTab(2, false);
 tabObj.enableTab(3, false);
 tabObj.select(0);
 }
 }
]};
 alertDlg.dataBind();
}

```

```

 alertDlg.hide();
 }
 function trainSelected(args) {
 selectedTrain = args.data;
 }
 function filterTrains(args) {
 /* Generating trains based on source and destination chosen */
 var result = [];
 var fromCity = startPoint.text;
 var toCity = endPoint.text;
 var count = Math.floor((Math.random() * 3) + 2);
 for (var i = 0; i < count; i++) {
 var details = [];
 details.TrainNo = Math.floor((Math.random() * 20) + 19000);
 details.Name = "Train " + i;
 details.Departure = fromCity;
 details.Arrival = toCity;
 details.Availability = Math.floor((Math.random() * 20) + 20);
 result.push(details);
 }
 availTrainGrid.dataSource = result;
 }
 function finalizeDetails(args) {
 /* Get the passenger details and update table with name and other details for confirmation */
 var reserved = [];
 var passCount = 0;
 for (var i = 1; i <= 3; i++) {
 var name = document.getElementById("pass_name" + i);
 var berthSelected = document.getElementById("pass_berth" + i);
 var gender = document.getElementById("pass_gender" + i);
 if (name.value !== "") {
 var details = [];
 var berth = berthSelected.value;
 details.TrainNo = selectedTrain.TrainNo.toString();
 details.PassName = name.value;
 details.Gender = gender.value;
 details.Berth = (berth === "") ? "Any" : berth;
 reserved.push(details);
 passCount++;
 }
 }
 var calcFare = 0;
 for (var i in cities) {
 if (startPoint.value == cities[i].name)
 calcFare = calcFare + cities[i].fare;
 if (endPoint.value == cities[i].name)
 calcFare = calcFare + cities[i].fare;
 }
 var displayAmt = document.getElementById("amount");
 if (ticketType.value === "Economy Class") {
 displayAmt.innerHTML = "Total payable amount: $" + passCount
* (300 + calcFare)
 } else if (ticketType.value === "Business Class") {
 displayAmt.innerHTML = "Total payable amount: $" + passCount
* (500 + calcFare)
 } else if (ticketType.value === "Common Class") {

```

```

 displayAmt.innerText = "Total payable amount: $" + passCount
 * (150 + calcFare)
 }
}
ticketDetailGrid.dataSource = reserved;
}
function tabNavigations(args) {
 switch (args.target.id) {
 case "searchNext":
 /* Validate the Source, Destination, Date and Class chosen
 and proceed only if all the fields are selected */
 if (!ej.base.isNullOrUndefined(startPoint.value) &&
 !ej.base.isNullOrUndefined(endPoint.value) &&
 !ej.base.isNullOrUndefined(ticketType.value) &&
 !ej.base.isNullOrUndefined(journeyDate.value)) {
 if (!ej.base.isNullOrUndefined(startPoint.value) &&
 startPoint.value === endPoint.value) {
 document.getElementById("err1").innerText = "*
Arrival point can't be same as Departure";
 }
 else {
 tabObj.enableTab(0, false);
 tabObj.enableTab(1, true);
 filterTrains(args);
 tabObj.select(1);
 document.getElementById("err1").innerText = "";
 document.getElementById("err2").innerText = "";
 }
 }
 else {
 document.getElementById("err1").innerText = "* Please
fill all the details before proceeding";
 }
 break;
 case "bookTickets":
 /* Based on the selected station generate Grid content to
 display trains available */
 if (availTrainGrid.getSelectedRecords() === undefined ||
 availTrainGrid.getSelectedRecords().length === 0) {
 document.getElementById("err2").innerText = "* Select
your convenient train";
 }
 else {
 tabObj.enableTab(2, true);
 tabObj.select(2);
 tabObj.enableTab(1, false);
 document.getElementById("err2").innerText = "";
 }
 break;
 case "confirmTickets":
 /* Get the Passenger details and validate the fields must
 not be left empty */
 var name = document.getElementById("pass_name1");
 var age = document.getElementById("pass_age1");
 var gender = document.getElementById("pass_gender1");
 if (name.value === "" || age.value === "" || gender.value
 === "") {

```



```

 document.getElementById("err3").innerText = "* Please
enter passenger details";
 }
 else {
 tabObj.enableTab(3, true);
 tabObj.select(3);
 tabObj.enableTab(2, false);
 document.getElementById("err3").innerText = "";
 finalizeDetails(args);
 }
 break;
case "makePayment":
 alertDlg.show();
 break;
case "goToSearch":
 /* Go back to change class, date or boarding places */
 selectedTrain = [];
 tabObj.enableTab(0, true);
 tabObj.select(0);
 tabObj.enableTab(1, false);
 break;
case "goBackToBook":
 /* Change the preferred train chosen already */
 tabObj.enableTab(1, true);
 tabObj.select(1);
 tabObj.enableTab(2, false);
 break;
case "goBackDetails":
 /* Update passenger detail before confirming the payment */
 tabObj.enableTab(2, true);
 tabObj.select(2);
 tabObj.enableTab(3, false);
 break;
 }
}
</script>
<style>
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 }
 #amount {
 text-align: right;
 font-size: 15px;
 padding: 15px 0px;
 }
 #passenger-table th {
 text-align: center;
 font-size: 14px;
 font-weight: 400;
 border: 1px solid gainsboro;
 }
 #passenger-table td,
 th {
 padding: 10px;
 }
 #passenger-table td {

```

```
 border: 1px solid gainsboro;
 }
 .name-header {
 width: 200px;
 }
 .age-header {
 width: 80px;
 }
 .gender-header {
 width: 120px;
 }
 .type-header {
 width: 150px;
 }
 .btn-container {
 text-align: center;
 }
 .search-item {
 padding-right: 50px;
 padding-bottom: 20px;
 }
 .item-title {
 font-weight: 500;
 padding-top: 10px;
 }
 @@media (max-width: 450px) {
 .e-sample-resize-container {
 height: 450px;
 }
 .responsive-align {
 width: 75%;
 margin: 0 auto;
 }
 .search-item {
 padding: 0 0 20px 0;
 width: 100%;
 }
 }
 #err1,
 #err2,
 #err3 {
 font-weight: bold;
 color: red;
 display: block;
 margin-top: 15px;
 }
 .wizard-title {
 font-size: 15px;
 }
 #PassengersList {
 overflow: auto;
 }
 #passenger-table {
 min-width: 500px;
 width: 100%;
 }
 .e-tab-section {
```

```
padding: 0 15px;
}
</style>
```

## WIZARD.CS

```
public ActionResult Index()
{
 List<DataFields> quotaData = new List<DataFields>();
 List<DataFields> genderData = new List<DataFields>();
 List<DataFields> berthData = new List<DataFields>();
 List<CitiesFields> citiesData = new List<CitiesFields>();
 quotaData.Add(new DataFields { ID = "1", Text = "Business Class" });
 quotaData.Add(new DataFields { ID = "2", Text = "Economy Class" });
 quotaData.Add(new DataFields { ID = "3", Text = "Common Class" });
 genderData.Add(new DataFields { ID = "1", Text = "Male" });
 genderData.Add(new DataFields { ID = "2", Text = "Female" });
 berthData.Add(new DataFields { ID = "1", Text = "Upper" });
 berthData.Add(new DataFields { ID = "2", Text = "Lower" });
 berthData.Add(new DataFields { ID = "3", Text = "Middle" });
 berthData.Add(new DataFields { ID = "4", Text = "Window" });
 berthData.Add(new DataFields { ID = "5", Text = "Aisle" });
 citiesData.Add(new CitiesFields { Name = "Chicago", Fare = 300 });
 citiesData.Add(new CitiesFields { Name = "San Francisco", Fare = 125 });
 citiesData.Add(new CitiesFields { Name = "Los Angeles", Fare = 175 });
 citiesData.Add(new CitiesFields { Name = "Seattle", Fare = 250 });
 citiesData.Add(new CitiesFields { Name = "Florida", Fare = 150 });
 ViewBag.headerTextOne = new TabHeader { Text = "New Booking" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Train List" };
 ViewBag.headerTextThree = new TabHeader { Text = "Add Passenger" };
 ViewBag.headerTextFour = new TabHeader { Text = "Make Payment" };
 ViewBag.quota = quotaData;
 ViewBag.gender = genderData;
 ViewBag.berth = berthData;
 ViewBag.citiesData = citiesData;
 ViewBag.content1 = "#booking";
 ViewBag.content2 = "#selectTrain";
 ViewBag.content3 = "#details";
 ViewBag.content4 = "#confirm";
 ViewBag.min = DateTime.Now;
 ViewBag.max = DateTime.Now.AddMonths(3);
 return View();
}

public class DataFields
{
 public string ID { get; set; }
 public string Text { get; set; }
}

public class CitiesFields
{
 public string Name { get; set; }
 public int Fare { get; set; }
}
```

### Load tab with DataSource

In the below demo, Data is fetched from an OData service using DataManager. The result data is formatted as a JSON object with header and content fields, which is set to items property of Tab.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div id="ej2Tab"></div>
<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
 .container {
 min-width: 350px;
 max-width: 500px;
 margin: 0 auto;
 }
</style>
<script type="text/javascript">
 ej.base.enableRipple(true);
 var itemsData = [];
 var mapping = { header: 'FirstName', content: 'Notes' };
 const SERVICE_URI =
'https://js.syncfusion.com/ejServices/Wcf/Northwind.svc/Employees';
 new ej.data.DataManager({ url: SERVICE_URI, adaptor: new
ej.data.ODataAdaptor })
 .executeQuery(new ej.data.Query().range(1, 4)).then((e) => {
 var result = e.result;
 for (var i = 0; i < result.length; i++) {
 itemsData.push({ header: { text: result[i][mapping.header]
}, content: result[i][mapping.content] });
 }
 //Initialize Tab component
 var tabObj = new ej.navigations.Tab({
 items: itemsData
 });
 //Render initialized Tab component
 tabObj.appendTo('#ej2Tab');
 });
</script>
```

#### DATA.CS

```
public ActionResult Index()
{
 return View();
}
```

### Add nested Tabs

Tab supports to render the nested level of Tabs by using `content` property. You can add the nested Tab element inside the parent Tab `content` property. To render the nested Tab, initialize the component using the id of Tab from a selected event handler.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Tab("ej2Tab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextOne, Content = "<div id='usa_tab'></div>" },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content = "<div id='france_tab'></div>" },
 new TabTabItem { Header = ViewBag.headerTextThree, Content = "<div id='australia_tab'></div>" }
 })
 .Selected("handleSelectEvent")
 .Created("handleCreatedEvent")
 .Render()
)
<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
 .container {
 min-width: 350px;
 max-width: 500px;
 margin: 0 auto;
 }
</style>
<script type="text/javascript">
 function handleCreatedEvent() {
 if (ej.base.isNullOrUndefined(document.querySelector('#usa_tab.e-tab')) {
 var usa_obj = new ej.navigations.Tab({
 items: [
 {

```

```

 header: { 'text': 'New York' },
 content: 'New York City comprises 5 boroughs sitting
where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a
densely populated borough that's among the world's major commercial,
financial and cultural centers. Its iconic sites include skyscrapers such as
the Empire State Building and sprawling Central Park. Broadway theater is
staged in neon-lit Times Square.'
 },
 {
 header: { 'text': 'Los Angeles' },
 content: 'Los Angeles is a sprawling Southern
California city and the center of the nation's film and television industry.
Near its iconic Hollywood sign, studios such as Paramount Pictures,
Universal and Warner Brothers offer behind-the-scenes tours. On Hollywood
Boulevard, TCL Chinese Theatre displays celebrities' hand- and footprints,
the Walk of Fame honors thousands of luminaries and vendors sell maps to
stars' homes.'
 },
 {
 header: { 'text': 'Chicago' },
 content: 'Chicago, on Lake Michigan in Illinois, is
among the largest cities in the U.S. Famed for its bold architecture, it has
a skyline punctuated by skyscrapers such as the iconic John Hancock Center,
1,451-ft. Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune
Tower. The city is also renowned for its museums, including the Art
Institute of Chicago with its noted Impressionist and Post-Impressionist
works.'
 }
]
});
usa_obj.appendTo('#usa_tab');
}

function handleSelectEvent(e) {
 if (e.selectedIndex === 0 &&
ej.base.isNullOrUndefined(document.querySelector('#usa_tab.e-tab'))) {
 var usa_obj = new ej.navigations.Tab({
 items: [
 {
 header: { 'text': 'New York' },
 content: 'New York City comprises 5 boroughs sitting
where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a
densely populated borough that's among the world's major commercial,
financial and cultural centers. Its iconic sites include skyscrapers such as
the Empire State Building and sprawling Central Park. Broadway theater is
staged in neon-lit Times Square.'
 },
 {
 header: { 'text': 'Los Angeles' },
 content: 'Los Angeles is a sprawling Southern
California city and the center of the nation's film and television industry.
Near its iconic Hollywood sign, studios such as Paramount Pictures,
Universal and Warner Brothers offer behind-the-scenes tours. On Hollywood
Boulevard, TCL Chinese Theatre displays celebrities' hand- and footprints,
the Walk of Fame honors thousands of luminaries and vendors sell maps to
stars' homes.'
 },
],
 });
 }
}

```

```

 {
 header: { 'text': 'Chicago' },
 content: 'Chicago, on Lake Michigan in Illinois, is
among the largest cities in the U.S. Famed for its bold architecture, it has
a skyline punctuated by skyscrapers such as the iconic John Hancock Center,
1,451-ft. Willis Tower (formerly the Sears Tower) and the neo-Gothic Tribune
Tower. The city is also renowned for its museums, including the Art
Institute of Chicago with its noted Impressionist and Post-Impressionist
works.'
 }
]
 });
 usa_obj.appendTo('#usa_tab');
} else if (e.selectedIndex === 1 &&
ej.base.isNullOrUndefined(document.querySelector('#france_tab.e-tab'))) {
 var france_obj = new ej.navigations.Tab({
 items: [
 {
 header: { 'text': 'Paris' },
 content: 'Paris, France capital, is a major European
city and a global center for art, fashion, gastronomy and culture. Its 19th-
century cityscape is crisscrossed by wide boulevards and the River Seine.
Beyond such landmarks as the Eiffel Tower and the 12th-century, Gothic
Notre-Dame cathedral, the city is known for its cafe culture and designer
boutiques along the Rue du Faubourg Saint-Honoré.'
 },
 {
 header: { 'text': 'Marseille' },
 content: 'Marseille, a port city in southern France,
has been a crossroads of immigration and trade since its founding by the
Greeks circa 600 B.C. At its heart is the Vieux-Port (Old Port), where
fishmongers sell their catch along the boat-lined quay. Basilique Notre-
Dame-de-la-Garde is a Romanesque-Byzantine church. Modern landmarks include
Le Corbusier's influential Cité Radieuse complex and Zaha Hadid's CMA CGM
Tower.'
 },
 {
 header: { 'text': 'Lyon' },
 content: 'Lyon, the capital city in France's
Auvergne-Rhône-Alpes region, sits at the junction of the Rhône and Saône
rivers. Its center reflects 2,000 years of history from the Roman
Amphithéâtre des Trois Gaules, medieval and Renaissance architecture in
Vieux (Old) Lyon, to the modern Confluence district on Presquîle peninsula.
Traboules, covered passageways between buildings, connect Vieux Lyon and La
Croix-Rousse hill.'
 }
]
 });
 france_obj.appendTo('#france_tab');
} else if (e.selectedIndex === 2 &&
ej.base.isNullOrUndefined(document.querySelector('#australia_tab.e-tab'))) {
 var australia_obj = new ej.navigations.Tab({
 items: [
 {
 header: { 'text': 'Sydney' },
 content: 'Sydney, capital of New South Wales and one
of Australia largest cities, is best known for its harbourfront Sydney Opera

```

House, with a distinctive sail-like design. Massive Darling Harbour and the smaller Circular Quay port are hubs of waterside life, with the arched Harbour Bridge and esteemed Royal Botanic Garden nearby. Sydney Tower's outdoor platform, the Skywalk, offers **360**-degree views of the city and suburbs.'

```

 },
 {
 header: { 'text': 'Melbourne' },
 content: 'Melbourne is the coastal capital of the
southeastern Australian state of Victoria. At the city centre is the modern
Federation Square development, with plazas, bars, and restaurants by the
Yarra River. In the Southbank area, the Melbourne Arts Precinct is the site
of Arts Centre Melbourne - a performing arts complex - and the National
Gallery of Victoria, with Australian and indigenous art.'
 },
 {
 header: { 'text': 'Brisbane' },
 content: 'Brisbane, capital of Queensland, is a
large city on the Brisbane River. Clustered in its South Bank cultural
precinct are the Queensland Museum and Sciencentre, with noted interactive
exhibitions. Another South Bank cultural institution is Queensland Gallery
of Modern Art, among Australia major contemporary art museums. Looming over
the city is Mt. Coot-tha, site of Brisbane Botanic Gardens.'
 }
]
});
australia_obj.appendTo('#australia_tab');
}
</script>

```

## NESTED.CS

```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "USA" };
 ViewBag.headerTextTwo = new TabHeader { Text = "France" };
 ViewBag.headerTextThree = new TabHeader { Text = "Australia" };
 return View();
}

```

## Set state persistence of the Tab Control

State persistence allows the Tab to retain the current modal value in the browser cookies for state maintenance. This action is handled through the enablePersistence property which is set to false by default. When it is set to true, some of the Tab component model values will be retained even after refreshing the page.

The following sample demonstrates how to set state persistence of the Tab component.

## CSHTML

```

@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Tab("ej2Tab")
 .Width("600px")
 .EnablePersistence(true)

```



```

 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "Twitter is
an online social networking service that enables users to send and read
short 140-character messages called tweets. Registered users can read and
post tweets, but those who are unregistered can only read them. Users access
Twitter through the website interface, SMS or mobile device app Twitter Inc.
is based in San Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone,
and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, with more than 100 million users posting 340 million
tweets a day in 2012.The service also handled 1.6 billion search queries per
day." },
 new TabBarItem { Header = ViewBag.headerText1, Content = "Facebook
is an online social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College roommates and fellow students Eduardo Saverin,
Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders had
initially limited the website's membership to Harvard students, but later
expanded it to colleges in the Boston area, the Ivy League, and Stanford
University. It gradually added support for students at various other
universities and later to high-school students." },
 new TabBarItem { Header = ViewBag.headerText2, Content = "WhatsApp
Messenger is a proprietary cross-platform instant messaging client for
smartphones that operates under a subscription business model. It uses the
Internet to send text messages, images, video, user location and audio media
messages to other users using standard cellular mobile numbers. As of
February 2016, WhatsApp had a user base of up to one billion,[10] making it
the most globally popular messaging application. WhatsApp Inc., based in
Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion." }
 })
 .Render()
)

```

### STYLES.CS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "Twitter" };
 ViewBag.headerText1 = new TabHeader { Text = "Facebook" };
 ViewBag.headerText2 = new TabHeader { Text = "WhatsApp" };
 return View();
}

```

Output be like the below.

TWITTER      FACEBOOK      WHATSAPP

Facebook is an online social networking service headquartered in Menlo Park, California. Its website was launched on February 4, 2004, by Mark Zuckerberg with his Harvard College roommates and fellow students Eduardo Saverin, Andrew McCollum, Dustin Moskovitz and Chris Hughes. The founders had initially limited the website's membership to Harvard students, but later expanded it to colleges in the Boston area, the Ivy League, and Stanford University. It gradually added support for students at various other universities and later to high-school students.

### Set custom animation

Tab supports custom animations for both previous and next actions from the provided animation option of `Animation` library. The animation property also allows you to set easing, duration, and various other effects.

Default animation is given as `SlideLeftIn` for previous tab animation and `SlideRightIn` for next tab animation. You can also disable the animation by setting the animation effect as `None`. Also, use the following CSS to disable indicator animation for animation effect as `None`.

`CSS

```
.e-tab .e-tab-header:not(.e-vertical) .e-indicator, .e-tab .e-tab-header.e-vertical .e-indicator {
transition: none;
}
`
```

The sample demonstrates some types of animation that suits Tab. You can check all the animation effects here.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class='row'>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Previous Animation </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 @Html.EJS().DropDownList("previous").DataSource(ViewBag.animationData).Index
 (0).Change("previousChange").Render()
 </div>
</div>
<div class='row'>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Next Animation </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 @Html.EJS().DropDownList("next").DataSource(ViewBag.animationData).Index(1).
 Change("nextChange").Render()
 </div>
</div>

@Html.EJS().Tab("ej2Tab").Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerText0, Content = "Twitter is an
online social networking service that enables users to send and read short
140-character messages called tweets. Registered users can read and post
tweets, but those who are unregistered can only read them. Users access
Twitter through the website interface, SMS or mobile device app Twitter Inc.
is based in San Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone,
and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, with more than 100 million users posting 340 million
tweets a day in 2012.The service also handled 1.6 billion search queries per
day." },
```

```

 new TabTabItem { Header = ViewBag.headerText1, Content = "Facebook is an
online social networking service headquartered in Menlo Park, California.
Its website was launched on February 4, 2004, by Mark Zuckerberg with his
Harvard College roommates and fellow students Eduardo Saverin, Andrew
McCollum, Dustin Moskovitz and Chris Hughes.The founders had initially
limited the website's membership to Harvard students, but later expanded it
to colleges in the Boston area, the Ivy League, and Stanford University. It
gradually added support for students at various other universities and later
to high-school students." },
 new TabTabItem { Header = ViewBag.headerText2, Content = "WhatsApp
Messenger is a proprietary cross-platform instant messaging client for
smartphones that operates under a subscription business model. It uses the
Internet to send text messages, images, video, user location and audio media
messages to other users using standard cellular mobile numbers. As of
February 2016, WhatsApp had a user base of up to one billion,[10] making it
the most globally popular messaging application. WhatsApp Inc., based in
Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion." }
 }).Render()
<script type="text/javascript">
 function previousChange(e) {
 var tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 tabObj.animation.previous.effect = e.value;
 }
 function nextChange(e) {
 var tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 tabObj.animation.next.effect = e.value;
 }
</script>

```

### STYLES.CSS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "Twitter" };
 ViewBag.headerText1 = new TabHeader { Text = "Facebook" };
 ViewBag.headerText2 = new TabHeader { Text = "WhatsApp" };
 ViewBag.animationData = new string[] { "SlideLeftIn", "SlideRightIn",
"FadeIn", "FadeOut", "FadeZoomIn", "FadeZoomOut", "ZoomIn", "ZoomOut",
"None" };
 return View();
}

```

Output be like the below.

|                    |              |
|--------------------|--------------|
| Previous Animation | SlideLeftIn  |
| Next Animation     | SlideRightIn |

**TWITTER**    **FACEBOOK**    **WHATSAPP**

Twitter is an online social networking service that enables users to send and read short 140-character messages called tweets. Registered users can read and post tweets, but those who are unregistered can only read them. Users access Twitter through the website interface, SMS or mobile device app. Twitter Inc. is based in San Francisco and has more than 25 offices around the world. Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone, and Noah Glass and launched in July 2006. The service rapidly gained worldwide popularity, with more than 100 million users posting 340 million tweets a day in 2012. The service also handled 1.6 billion search queries per day.

### Load Tab items dynamically

Tabs can be added dynamically by passing array of items and index value to the addTab method.

In the following demo, you can add the tab content by clicking the +. Enter the new Tab heading and content details in the available text boxes, click 'Add Tab' button to pass the details as an array and here last index is calculated to append the new tab at the end.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div id="tab1_content" style="display: none">

 Click on the "+" header to add dynamic tab items.
 It displays input elements to get the new tab information.
 Add details and click the "Add Tab" button to open the newly
added tab.

</div>
<div id="form-container" style="display: none">
 <div class="e-float-input">
 <input type="text" id="tab-title" required="" />

 <label class="e-float-text">Enter header title</label>
 </div>

 <div class="e-float-input">
 <textarea rows="5" type="text" id="tab-content"
required=""></textarea>

 <label class="e-float-text">Enter content</label>
 </div>

 <div class="btn-section">
 <button id="btn-add" class="btn btn-default"
onclick="btnClicked(this)">Add Tab</button>

 * Title is mandatory to add a new Tab
 </div>
</div>
@(Html.EJS().Tab("ej2Tab")
.Items(new List<TabBarItem> {
```

```

 new TabTabItem { Header = ViewBag.headerTextOne, Content =
"#tabl_content" },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content = "#form-
container" }
 })
 .Created("tabCreated")
 .Selected("tabSelected")
 .Render()
)
<style>
#container {
 visibility: hidden;
}
#loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
}
.e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
}
.container {
 min-width: 350px;
 max-width: 500px;
 margin: 0 auto;
}
#form-container {
 margin: 0 auto;
 max-width: 300px;
}
.btn-section {
 text-align: center;
}
.add-tab-btn-section td {
 padding: 10px;
}
.info {
 font-weight: bold;
}
.e-add-icon::before {
 content: '\e7d5';
}
</style>
<script type="text/javascript">
function tabCreated() {
 var totalItems = 0;
 var addBtn = document.querySelectorAll(".e-ileft.e-icon");
 addBtn[0].setAttribute("title", "Add Tab");
}
function tabSelected(args) {
 if (args.selectedIndex === document.querySelectorAll('#ej2Tab .e-
toolbar-item').length - 1) {

```

```

 document.getElementById('tab-title').value = '';
 document.getElementById('tab-content').value = '';
 }
}
function btnClicked(e) {
 var title = document.getElementById('tab-title').value;
 var content = document.getElementById('tab-content').value;
 var tabObj = document.getElementById("ej2Tab").ej2_instances[0];
 // Required tab item object formed by using textbox inputs
 var item = { header: { text: title }, content:
ej.base.createElement('pre', { innerHTML: content.replace(/\n/g, '
\n')
}).outerHTML };
 totalItems = document.querySelectorAll('#ej2Tab .e-toolbar-
item').length;
 // Item object and the index argument passed into the addTab method
to add a new tab
 tabObj.addTab([item], totalItems - 1);
}
</script>

```

## DYNAMIC.CS

```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "USA" };
 ViewBag.headerTextTwo = new TabHeader { IconCss = "e-add-icon" };
 return View();
}

```

## Create collapsible Tabs

You can achieve collapse and expand functionality in Tab by adding/removing a custom CSS class in the click event handler for each tab.

- Define a CSS class to set the style property display as none. Here 'collapse' class is added to the content element for hiding it.
- Bind the `select` event for Tab to collapse the initially selected Tab item and bind custom click handler for the Tab headers.
- In the event handler, add and remove 'collapse' class to hide and show the corresponding Tab content.

## CSHTML

```

@using Syncfusion.EJ2.Navigations;
@{
 var content0 = "Twitter is an online social networking service that
enables users to send and read short 140-character" +
 "messages called 'tweets'.Registered users can read and
post tweets, but those who are unregistered can only read" +
 "them.Users access Twitter through the website interface,
SMS or mobile device app Twitter Inc. is based in San" +
 "Francisco and has more than 25 offices around the
world.Twitter was created in March 2006 by Jack Dorsey," +

```

```

 "Evan Williams, Biz Stone, and Noah Glass and launched in
 July 2006. The service rapidly gained worldwide popularity," +
 "with more than 100 million users posting 340 million
 tweets a day in 2012.The service also handled 1.6 billion" +
 "search queries per day.";
 var content1 = "Facebook is an online social networking service
 headquartered in Menlo Park, California. Its website was" +
 "launched on February 4, 2004, by Mark Zuckerberg with
 his Harvard College roommates and fellow students Eduardo" +
 "Saverin, Andrew McCollum, Dustin Moskovitz and Chris
 Hughes.The founders had initially limited the websites" +
 "membership to Harvard students, but later expanded it to
 colleges in the Boston area, the Ivy League, and Stanford" +
 "University.It gradually added support for students at
 various other universities and later to high-school students.";
 var content2 = "WhatsApp Messenger is a proprietary cross-platform
 instant messaging client for smartphones that operates" +
 "under a subscription business model.It uses the Internet
 to send text messages, images, video, user location and" +
 "audio media messages to other users using standard
 cellular mobile numbers.As of February 2016, WhatsApp had a user" +
 "base of up to one billion,[10] making it the most
 globally popular messaging application.WhatsApp Inc., based in" +
 "Mountain View, California, was acquired by Facebook
 Inc.on February 19, 2014, for approximately US$19.3 billion.";
 }
 <div class="info">
 Collapsible Tabs
 </div>

 <i>The active tab can be toggled to expand and collapse its content.</i>

 @(Html.EJS().Tab("ej2Tab")
 .Items(new List<TabTabItem> {
 new TabTabItem { Header = ViewBag.headerTextOne, Content = @content0
 },
 new TabTabItem { Header = ViewBag.headerTextTwo, Content = @content1
 },
 new TabTabItem { Header = ViewBag.headerTextThree, Content =
 @content2 }
 })
 .CssClass("e-background")
 .Created("tabCreated")
 .Selected("tabSelected")
 .Render()
)
 <style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;

```

```

 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
 .container {
 min-width: 350px;
 margin: 0 10px;
 }
 .info {
 margin: 10px;
 font-weight: bold;
 }
 #ej2Tab.e-tab .e-content > .e-item.e-active.collapse {
 display: none;
 }
 #ej2Tab.e-tab .e-tab-header .e-indicator.collapse {
 display: none;
 }
</style>
<script type="text/javascript">
 var trgIndex;
 var actLine;
 var target;
 function tabCreated() {
 actLine = document.querySelector('#ej2Tab.e-tab .e-indicator');
 target = document.querySelector('#ej2Tab.e-tab .e-content .e-item.e-
active');
 target.classList.add('collapse');
 actLine.classList.add('collapse');
 }
 function tabSelected(e) {
 var cnttrgs = document.querySelectorAll('#ej2Tab.e-tab > .e-content
> .e-item');
 for (var i = 0; i < cnttrgs.length; i++) {
 cnttrgs[i].classList.remove('collapse');
 }
 if (actLine !== undefined) {
 actLine.classList.remove('collapse');
 }
 trgIndex = e.selectedIndex;
 // Custom click event binding for each tab item to make
collapse/expand
 e.selectedItem.addEventListener('click', function (e) {
 updateCollapseClass();
 });
 }
 function updateCollapseClass() {
 // Custom classes are added/removed from tab content and active line
element, when the same tab item again clicked
 var cntEle = document.querySelector('#ej2Tab.e-tab .e-content .e-
item.e-active');
 if (cntEle.classList.contains('collapse')) {
 cntEle.classList.remove('collapse');
 actLine.classList.remove('collapse');
 }
 }

```



```

 else {
 cntEle.classList.add('collapse');
 actLine.classList.add('collapse');
 }
 }
}
</script>

```

### COLLAPSIBLE.CS

```

public ActionResult Index()
{
 ViewBag.headerTextOne = new TabHeader { Text = "Twitter" };
 ViewBag.headerTextTwo = new TabHeader { Text = "Facebook" };
 ViewBag.headerTextThree = new TabHeader { Text = "WhatsApp" };
 return View();
}

```

### Customize Tab content height

You can change the Tab content height by using the `heightAdjustMode` property. By default, the Tab content `heightAdjustMode` property is set to `Content` value.

- **None:** Each tab content height is set based on the Tab height. This value is used only the tab component having the `height` property.
- **Auto:** Each tab content height will take the maximum height of all other tabs content.
- **Content:** Each tab content height is set based on their own content.
- **Fill:** Each tab content height is set based on the full height of Tabs parent element.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class='row'>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">
 <label> Height Adjust Mode </label>
 </div>
 <div class="col-xs-6 col-sm-6 col-lg-6 col-md-6">

@Html.EJS().DropDownList("contentHeight").DataSource(ViewBag.heightData).Index(1).Change("onChange").Render()
 </div>
</div>
@ (Html.EJS().Tab("ej2Tab")
 .Height("400px")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "Twitter is
an online social networking service that enables users to send and read
short 140-character messages called tweets. Registered users can read and
post tweets, but those who are unregistered can only read them. Users access
Twitter through the website interface, SMS or mobile device app Twitter Inc.
is based in San Francisco and has more than 25 offices around the world.
Twitter was created in March 2006 by Jack Dorsey, Evan Williams, Biz Stone,
and Noah Glass and launched in July 2006. The service rapidly gained
worldwide popularity, with more than 100 million users posting 340 million

```

```

tweets a day in 2012.The service also handled 1.6 billion search queries per
day." },
 new TabBarItem { Header = ViewBag.headerText1, Content = "Facebook
is an online social networking service headquartered in Menlo Park,
California. Its website was launched on February 4, 2004, by Mark Zuckerberg
with his Harvard College roommates and fellow students Eduardo Saverin,
Andrew McCollum, Dustin Moskovitz and Chris Hughes.The founders had
initially limited the website's membership to Harvard students, but later
expanded it to colleges in the Boston area, the Ivy League, and Stanford
University. It gradually added support for students at various other
universities and later to high-school students." },
 new TabBarItem { Header = ViewBag.headerText2, Content = "WhatsApp
Messenger is a proprietary cross-platform instant messaging client for
smartphones that operates under a subscription business model. It uses the
Internet to send text messages, images, video, user location and audio media
messages to other users using standard cellular mobile numbers. As of
February 2016, WhatsApp had a user base of up to one billion,[10] making it
the most globally popular messaging application. WhatsApp Inc., based in
Mountain View, California, was acquired by Facebook Inc. on February 19,
2014, for approximately US$19.3 billion." }
 })
 .Render()
)

</div>
<script type="text/javascript">
 function onChange(e) {
 var tabObj = document.getElementById("ej2Tab").ej2_instances[0];
 tabObj.heightAdjustMode = e.value;
 }
</script>

```

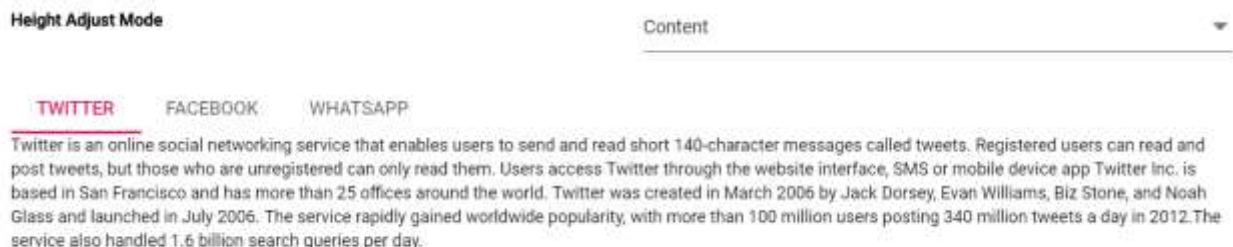
## HEIGHT.CS

```

public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "Twitter" };
 ViewBag.headerText1 = new TabHeader { Text = "Facebook" };
 ViewBag.headerText2 = new TabHeader { Text = "WhatsApp" };
 ViewBag.heightData = new string[] { "None", "Content", "Fill", "Auto" };
 return View();
}

```

Output be like the below.



### How to customize Tab scrollStep

Tab supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize **ScrollStep** property for scrolling distance. Refer to the following code example.

By using Tab scrollStep property, pass a required value to customize tab scrollStep.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().Tab("ej2Tab")
 .Width("600px")
 .ScrollStep("50")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language." },
 new TabBarItem { Header = ViewBag.headerText1, Content = "C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 5.0, which was released on August 15, 2012." },
 new TabBarItem { Header = ViewBag.headerText2, Content = "Java is a set of computer software and specifications developed by Sun Microsystems, later acquired by Oracle Corporation, that provides a system for developing application software and deploying it in a cross - platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While less common, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages." },
 new TabBarItem { Header = ViewBag.headerText3, Content = "The command-line compiler, VBC.EXE, is installed as part of the freeware .NET Framework SDK. Mono also includes a command - line VB.NET compiler. The most recent version is VB 2012, which was released on August 15, 2012." },
 new TabBarItem { Header = ViewBag.headerText4, Content = "Xamarin is a San Francisco, California based software company created in May 2011[3] by the engineers that created Mono, [4] Mono for Android and MonoTouch that are cross - platform implementations of the Common Language Infrastructure (CLI) and Common Language Specifications (often called Microsoft.NET). With a C#-shared codebase, developers can use Xamarin tools to write native Android, iOS, and Windows apps with native user interfaces and share code across multiple platforms.[5] Xamarin has over 1 million developers in more than 120 countries around the World as of May 2015." },
 new TabBarItem { Header = ViewBag.headerText5, Content = "ASP.NET is an open-source server-side web application framework designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications and web services. It was first released in January 2002 with version 1.0 of the .NET Framework, and is the successor to Microsoft Active Server Pages (ASP) technology. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language. The
```

```
ASP.NET SOAP extension framework allows ASP.NET components to process SOAP
messages." },
 new TabTabItem { Header = ViewBag.headerText6, Content = "The
ASP.NET MVC is a web application framework developed by Microsoft, which
implements the model-view-controller(MVC) pattern.It is open - source
software, apart from the ASP.NET Web Forms component which is proprietary.In
the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET Web
Pages(a platform using only Razor pages) will merge into a unified MVC 6.The
project is called ASP.NET vNext." },
 new TabTabItem { Header = ViewBag.headerText7, Content = "JavaScript
(JS) is an interpreted computer programming language. It was originally
implemented as part of web browsers so that client - side scripts could
interact with the user, control the browser, communicate asynchronously, and
alter the document content that was displayed.[5] More recently, however, it
has become common in both game development and the creation of desktop
applications." }
})
.Render()
}
```

## STYLES.CSS

```
public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp(C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
 ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
 ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
 ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
 ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
 return View();
}
```

## Populate Tab items and their content through ViewBag

For the Tab control, the tab items can be rendered in the controller and can be returned as ViewBag to bind as items. You can also map the content to other contents using the mapping id in controller to return as ViewBag. Refer to the below sample, which takes [chart](#), [grid](#), [calender](#) as its content through viewBag content id mapped in view.

## CSHTML

```
@using Syncfusion.EJ2.Navigations;
<div class="info">
 Tab content loaded from ViewBag
</div>
<div id="Content1" style="display: none">
 @(Html.EJS().Chart("ej2chart")
 .Width("100%")
 .Height("100%")
 .Series(series =>
 {
 series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line).
 Marker(mr => mr.Visible(true)).
```

```

 XName("x").
 YName("yValue").
 DataSource(ViewBag.dataSource).
 Name("Gold").
 Width(2).Add();
 })
 .PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
 .Title("Olympic Medal Counts - RIO")
 .Render()
)
</div>
<div id="Content2" style="display: none">
 @(Html.EJS().Grid("ej2grid")
 .DataSource(dataManger =>
 {
dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/Products")
.CrossDomain(true).Adaptor("ODataV4Adaptor");
 })
 .Columns(col =>
 {
 col.Field("ProductID").HeaderText("Product
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("ProductName").HeaderText("Product
Name").Width("150").Add();
 col.Field("UnitPrice").HeaderText("Supplier
ID").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("UnitsInStock").HeaderText("QuantityPerUnit").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Discontinued").HeaderText("Discontinued").Width("140").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Center).Type("boolean").DisplayAsCheckBox(true).Add();
 })
 .AllowPaging()
 .Render()
)
</div>
<div id="Content3" style="display: none">
 @(Html.EJS().Calendar("ej2calendar").Render())
</div>
 @(Html.EJS().Tab("ej2Tab").Items(ViewBag.items).Render())

```

### VIEWBAG.CS

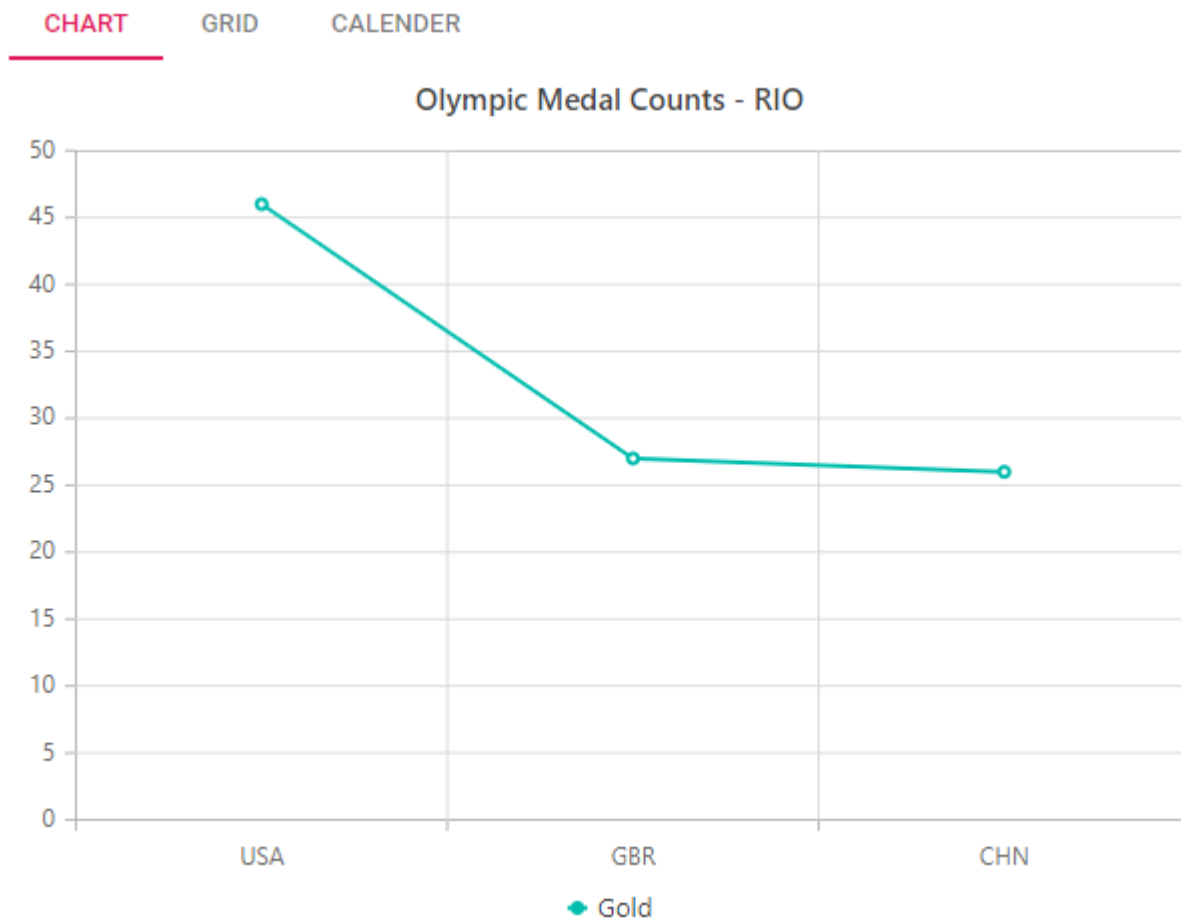
```

public ActionResult Index()
{
 List<TabTabItem> tabItems = new List<TabTabItem>();
 tabItems.Add(new TabTabItem { Header = new TabHeader { Text = "Chart" },
Content = "#Content1" });
 tabItems.Add(new TabTabItem { Header = new TabHeader { Text = "Grid" },
Content = "#Content2" });
 tabItems.Add(new TabTabItem { Header = new TabHeader { Text = "Calender"
}, Content = "#Content3" });
}

```

```
ViewBag.items = tabItems;
//chart data
List<ColumnChartData> chartData = new List<ColumnChartData>
{
 new ColumnChartData { x= "USA", yValue= 46 },
 new ColumnChartData { x= "GBR", yValue= 27 },
 new ColumnChartData { x= "CHN", yValue= 26 }
};
ViewBag.dataSource = chartData;
return View();
}
public class ColumnChartData
{
 public string x;
 public double yValue;
}
```

Output be like the below.



Render the Tab items using content template

You can bind any data in Tab items, by simply using the content template property in ASP.NET Tab.

In the below demo, the tab items are given as [chart](#), [grid](#), [calender](#) using the content template. In the content template you can give the header using e-tab-header and content using e-content class.

CSHTML

```

@using Syncfusion.EJ2.Navigations;
<div class="info">
 Content Template
</div>
@(Html.EJS().Tab("ej2Tab")
 .ContentTemplate(
 @<div>
 <div class="e-tab-header">
 <div>Grid</div>
 <div>Chart</div>
 <div>Calendar</div>
 </div>
 <div class="e-content">
 <div>
 @(Html.EJS().Grid("ej2grid")
 .Height("400px")
 .DataSource(dataManger => {
dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/Products")
.CrossDomain(true).Adaptor("ODataV4Adaptor");
 })
 .Columns(col => {
 col.Field("ProductID").HeaderText("Product
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("ProductName").HeaderText("Product
Name").Width("150").Add();
 col.Field("UnitPrice").HeaderText("Supplier
ID").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

 col.Field("UnitsInStock").HeaderText("QuantityPerUnit").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

 col.Field("Discontinued").HeaderText("Discontinued").Width("140").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Center).Type("boolean").DisplayAsCheckBox(true).Add();
 })
 .AllowPaging()
 .Render()
)
 </div>
 <div>
 @(Html.EJS().Chart("ej2chart")
 .Width("100%")
 .Height("100%")
 .Series(series => {
series.Type(Syncfusion.EJ2.Charts.ChartSeriesType.Line)
.Marker(mr => mr.Visible(true))
.XName("x")
.YName("yValue")
.DataSource(ViewBag.dataSource)
.Name("Gold")
.Width(2).Add();

```

```

 })
 .PrimaryXAxis(px =>
px.Interval(1).ValueType(Syncfusion.EJ2.Charts.ValueType.Category))
 .Title("Olympic Medal Counts - RIO")
 .Render()
)
</div>
<div>
 @Html.EJS().Calendar("ej2calendar").Render()
</div>
</div>
</div>
)
.HeightAdjustMode(HeightStyles.Content)
.OverflowMode(OverflowMode.Scrollable)
.Render()
)

```

### CONTENTTEMPLATE.CS

```

public ActionResult Index()
{
 return View();
}

```

Output be like the below.

GRID

CHART

CALENDAR

| Product ID | Product Name                    | Supplier ID | QuantityPerUnit | Discontinued                        |
|------------|---------------------------------|-------------|-----------------|-------------------------------------|
| 1          | Chai                            | 18          | 39              | <input type="checkbox"/>            |
| 2          | Chang                           | 19          | 17              | <input type="checkbox"/>            |
| 3          | Aniseed Syrup                   | 10          | 13              | <input type="checkbox"/>            |
| 4          | Chef Anton's Cajun Seasoning    | 22          | 53              | <input type="checkbox"/>            |
| 5          | Chef Anton's Gumbo Mix          | 21.35       | 0               | <input checked="" type="checkbox"/> |
| 6          | Grandma's Boysenberry Spread    | 25          | 120             | <input type="checkbox"/>            |
| 7          | Uncle Bob's Organic Dried Pears | 30          | 15              | <input type="checkbox"/>            |
| 8          | Northwoods Cranberry Sauce      | 40          | 6               | <input type="checkbox"/>            |
| 9          | Mishi Kobe Niku                 | 97          | 29              | <input checked="" type="checkbox"/> |
| 10         | Ikura                           | 31          | 31              | <input type="checkbox"/>            |
| 11         | Queso Cabrales                  | 21          | 22              | <input type="checkbox"/>            |

1 of 7 pages (77 items)

Load the content as partial view to Tab

Since Tab is a Navigation control, it doesn't have support to load any content directly or using any DataAdaptor. But it is provided with the items support. So to load the content as partial view, you would need to make use of the AJAX or EJ2 Datamanager as described in our [How-To](#) section help document.

In the below demo, we have explained on how to create the Tab items dynamically and then to load the other Syncfusion controls in it from partial views.



**CSHTML**

```

<div id='GridOrder1'></div>
<div id='GridOrder2'></div>
@Html.EJS().Tab("MainTab").Created("Created").Selecting("Selecting").Render(
)
<script>
 function Created() {
 var TabObj = document.getElementById("MainTab").ej2_instances[0];
 var ajax = new ej.base.Ajax('@Url.Action("PartialView1", "Home")',
'GET', true);
 ajax.send().then();
 ajax.onSuccess = function (data) {
 TabObj.addTab([{ header: { 'text': 'Grid1' }, content:
"#GridOrder1" }], 0);
 TabObj.addTab([{ header: { 'text': 'Grid2' }, content:
"#GridOrder2" }], 1);
 $("#GridOrder1").html(data);
 }
 }
 function Selecting(e) {
 if (e.selectingIndex != 0) {
 var ajax = new ej.base.Ajax('@Url.Action("PartialView2",
"Home")', 'GET', true);
 ajax.send().then();
 ajax.onSuccess = function (data) {
 $("#GridOrder2").html(data);
 }
 }
 }
</script>
//Code in the PartialView1
<h2>Grid</h2>
@ (Html.EJS().Grid("Grid1").Height(250)
 .DataSource(dataManger =>
 {

dataManger.Url("https://ej2services.syncfusion.com/production/web-
services/api/Orders").CrossDomain(true).Adaptor("WebApiAdaptor");
 })
 .Columns(col =>
 {
 col.Field("OrderID").HeaderText("Order
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("CustomerID").HeaderText("Customer
ID").Width("160").Add();
 col.Field("EmployeeID").HeaderText("Employee
ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

 col.Field("Freight").HeaderText("Freight").Width("150").Format("C2").TextAli
gn(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("ShipCountry").HeaderText("Ship
Country").Width("150").Add();
 }).AllowPaging().PageSettings(page => page.PageCount(3))
 .Render()
)
@Html.EJS().ScriptManager()

```

```
//Code in the PartialView2
<h2>Grid</h2>
@(Html.EJS().Grid("Grid2")
 .DataSource(dataManger =>
 {

dataManger.Url("https://services.odata.org/V4/Northwind/Northwind.svc/Products").CrossDomain(true).Adaptor("ODataV4Adaptor");
 })
 .Columns(col =>
 {
 col.Field("ProductID").HeaderText("Product ID").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("ProductName").HeaderText("Product Name").Width("150").Add();
 col.Field("UnitPrice").HeaderText("Supplier ID").Width("130").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("UnitsInStock").HeaderText("QuantityPerUnit").Width("120").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

col.Field("Discontinued").HeaderText("Discontinued").Width("140").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Center).Type("boolean").DisplayAsCheckBox(true).Add();
 })
 .AllowPaging()
 .Render()
)
@Html.EJS().ScriptManager()
```

## PARTIALVIEW.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public partial class TabController : Controller
 {
 // GET: //
 public IActionResult DefaultFunctionalities()
 {
 ViewBag.headerText0 = new TabHeader { Text = "Grid1" };
 ViewBag.headerText1 = new TabHeader { Text = "Grid2" };
 return View();
 }
 public ActionResult PartialView1()
 {
 return PartialView();
 }
 public ActionResult PartialView2()
 {
 return PartialView();
 }
 }
}
```

```

 }
 }
}

```

Output be like the below.

GRID1

GRID2

Grid

| Order ID | Customer ID | Employee ID | Freight | Ship Country |
|----------|-------------|-------------|---------|--------------|
| 10001    | ALFKI       | 1           | \$2.30  | Denmark      |
| 10002    | ANATR       | 3           | \$3.30  | Brazil       |
| 10003    | ANTON       | 2           | \$4.30  | Germany      |
| 10004    | BONP        | 4           | \$5.30  | Austria      |
| 10005    | BOLID       | 5           | \$6.30  | Switzerland  |
| 10006    | ALFKI       | 2           | \$4.60  | Denmark      |
| 10007    | ANATR       | 4           | \$6.60  | Brazil       |

1 of 4 pages (45 items)

GRID1

GRID2

Grid

| Product ID | Product Name                    | Supplier ID | QuantityPerUnit | Discontinued                        |
|------------|---------------------------------|-------------|-----------------|-------------------------------------|
| 1          | Chai                            | 18          | 39              | <input type="checkbox"/>            |
| 2          | Chang                           | 19          | 17              | <input type="checkbox"/>            |
| 3          | Aniseed Syrup                   | 10          | 13              | <input type="checkbox"/>            |
| 4          | Chef Anton's Cajun Seasoning    | 22          | 53              | <input type="checkbox"/>            |
| 5          | Chef Anton's Gumbo Mix          | 21.35       | 0               | <input checked="" type="checkbox"/> |
| 6          | Grandma's Boysenberry Sprig     | 25          | 120             | <input type="checkbox"/>            |
| 7          | Uncle Bob's Organic Dried Pears | 30          | 15              | <input type="checkbox"/>            |
| 8          | Northwoods Cranberry Sauce      | 40          | 6               | <input type="checkbox"/>            |
| 9          | Mishi Kobe Niku                 | 97          | 29              | <input checked="" type="checkbox"/> |
| 10         | Ikura                           | 31          | 31              | <input type="checkbox"/>            |
| 11         | Queso Cabrales                  | 21          | 22              | <input type="checkbox"/>            |

### How to prevent reorder active tab while selecting inside popup

We can able to prevent the changing of the active tab item on resizing the browser when overflow mode is popup by using the `reorderActiveTab` property. By default, the active Tab should be reordered when we click the tab items from the popup. If we set `false` to `reorderActiveTab` property the active tab item from the popup will not be reordered and an active item is highlighted inside the popup. The following code example depicts to prevent the reorder active tab item inside the popup.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
<div class="col-lg-12 control-section">
 <div id="default" class="e-sample-resize-container">

```

```

@Html.EJS().Tab("ej2Tab").HeightAdjustMode(HeightStyles.None).Height("150px"
).Width("700px").Items(ViewBag.adaptiveItems).OverflowMode(OverflowMode.Popu
p).ReorderActiveTab(false).Render()
</div>
</div>
<style>
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }

 #ej2Tab {
 margin-top: 80px;
 margin-left: 80px;
 }

 #default {
 margin-top: 15px;
 }
 #default .row {
 margin-right: 20px;
 margin-left: 20px;
 }
</style>
<script type="text/javascript">
 var tabObj;
 function tabCreated() {
 tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 }
 // Change event function for DropDownList component
 function changeHeaderPosition(e) {
 tabObj.headerPlacement = e.itemData.value;
 tabObj.dataBind();
 }
 function changeOverFlowMode(e) {
 tabObj.overflowMode = e.itemData.value;
 tabObj.dataBind();
 }
</script>

```

### REORDERACTIVETAB.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Navigations;
// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
namespace EJ2CoreSampleBrowser.Controllers
{
 public partial class TabController : Controller
 {

```

```

List<TabTabItem> adaptiveItems = new List<TabTabItem>();
public IActionResult ResponsiveModes()
{
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp(C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
 ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
 ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
 ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
 ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
 ViewBag.headerText8 = new TabHeader { Text = "React" };
 ViewBag.headerText9 = new TabHeader { Text = "Angular" };
 ViewBag.headerText10 = new TabHeader { Text = "Vue" };
 ViewBag.headerText11 = new TabHeader { Text = "Typescript" };
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "HTML" }, Content = "HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web pages.Along with
CSS, and JavaScript, HTML is a cornerstone technology, used by most websites
to create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages.HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "C Sharp(C#)" }, Content = "C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
is led by Anders Hejlsberg.The most recent version is C# 5.0, which was
released on August 15, 2012." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "Java" }, Content = "Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross - platform computing environment.Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers.While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "VB.Net" }, Content = "The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also includes a command - line
VB.NET compiler.The most recent version is VB 2012, which was released on
August 15, 2012." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "Xamarin" }, Content = "Xamarin is a San Francisco, California based
software company created in May 2011[3] by the engineers that created Mono,
[4] Mono for Android and MonoTouch that are cross - platform implementations
of the Common Language Infrastructure(CLI) and Common Language
Specifications(often called Microsoft.NET).With a C#-shared codebase,
developers can use Xamarin tools to write native Android, iOS, and Windows
apps with native user interfaces and share code across multiple
platforms.[5] Xamarin has over 1 million developers in more than 120
countries around the World as of May 2015." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "ASP.NET" }, Content = "ASP.NET is an open-source server-side web
application framework designed for web development to produce dynamic web
pages.It was developed by Microsoft to allow programmers to build dynamic

```

```

web sites, web applications and web services.It was first released in
January 2002 with version 1.0 of the.NET Framework, and is the successor to
Microsoft Active Server Pages (ASP) technology. ASP.NET is built on the
Common Language Runtime (CLR), allowing programmers to write ASP.NET code
using any supported .NET language. The ASP.NET SOAP extension framework
allows ASP.NET components to process SOAP messages." });
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "ASP.NET MVC" }, Content = "The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the model-view-controller(MVC)
pattern.It is open - source software, apart from the ASP.NET Web Forms
component which is proprietary.In the later versions of ASP.NET, ASP.NET
MVC, ASP.NET Web API, and ASP.NET Web Pages(a platform using only Razor
pages) will merge into a unified MVC 6.The project is called ASP.NET vNext."
});
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "JavaScript" }, Content = "JavaScript (JS) is an interpreted computer
programming language. It was originally implemented as part of web browsers
so that client - side scripts could interact with the user, control the
browser, communicate asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in both game
development and the creation of desktop applications." });
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "React" }, Content = "React is a free and open-source front-end JavaScript
library for building user interfaces based on UI components. It is
maintained by Meta and a community of individual developers and companies."
});
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "Angular" }, Content = "Angular is a TypeScript-based free and open-source
web application framework led by the Angular Team at Google and by a
community of individuals and corporations. Angular is a complete rewrite
from the same team that built AngularJS" });
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "Vue" }, Content = "Vue.js is an open-source model-view-viewmodel front
end JavaScript framework for building user interfaces and single-page
applications. It was created by Evan You, and is maintained by him and the
rest of the active core team members." });
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "Typescript" }, Content = "TypeScript is a programming language developed
and maintained by Microsoft. It is a strict syntactical superset of
JavaScript and adds optional static typing to the language. TypeScript is
designed for the development of large applications and transcompiles to
JavaScript." });
 ViewBag.adaptiveItems = adaptiveItems;
 return View();
}
}
}

```

### How to find whether the tab is selected programmatically or user interaction

We can able to find the tab selection whether it is selected by user interaction or programmatically way in the `selecting` and `selected` event argument with the field of `isInteracted`. When the user changes the tab through click actions it will return true otherwise, it will return false. The following code example depicts to find the tab selecting the state in selecting and selected events.

### CSHTML

```

@using Syncfusion.EJ2;
@using Syncfusion.EJ2.Navigations;
@using Syncfusion.EJ2.DropDowns;
<div class="col-lg-12 control-section">
 <div id="default" class="e-sample-resize-container">
 <div className="EventLog" id="EventLog"></div>
 @Html.EJS().DropDownList("Mode").Placeholder("Select Tab Item using
dropdown").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Always).Index
(0).Width("100
%").DataSource((IEnumerable<Object>)ViewBag.dropDownData).Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "Text", Value =
"Id" }).Change("dropdownChange").Render()

@Html.EJS().Tab("ej2Tab").HeightAdjustMode(HeightStyles.Auto).Height("150px"
).Width("700px").Items(ViewBag.adaptiveItems).Selecting("Selecting").Selecte
d("Selected").Render()
 </div>
</div>
<style>
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
 #ej2Tab {
 margin-top: 80px;
 margin-left: 80px;
 }
 #default {
 margin-top: 15px;
 }
 #default .row {
 margin-right: 20px;
 margin-left: 20px;
 }
</style>
<script type="text/javascript">
 var tabObj;
 // Change event function for DropDownList component
 function dropdownChange(e) {
 tabObj = document.getElementById('ej2Tab').ej2_instances[0];
 tabObj.select(e.value);
 }
 function Selecting(args) {
 getInteractionDetail(args.isInteracted);
 }
 function Selected(args) {
 getInteractionDetail(args.isInteracted);
 }
 function getInteractionDetail(interact) {
 let eventlog = interact
 ? 'Tab Item selected by user interaction'
 : 'Tab Item selected by programmatically';
 document.getElementById('EventLog').innerHTML =
document.getElementById('EventLog'
).innerHTML = '' + eventlog + '';
 }

```

```
</script>
```

## TABSELECTION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Syncfusion.EJ2.Navigations;
// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
namespace EJ2CoreSampleBrowser.Controllers
{
 public partial class TabController : Controller
 {
 List<TabTabItem> adaptiveItems = new List<TabTabItem>();
 public IActionResult ResponsiveModes()
 {
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp(C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
 ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
 ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
 ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
 ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "HTML" }, Content = "HyperText Markup Language, commonly referred to as
HTML, is the standard markup language used to create web pages.Along with
CSS, and JavaScript, HTML is a cornerstone technology, used by most websites
to create visually engaging web pages, user interfaces for web applications,
and user interfaces for many mobile applications.[1] Web browsers can read
HTML files and render them into visible or audible web pages.HTML describes
the structure of a website semantically along with cues for presentation,
making it a markup language, rather than a programming language." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "C Sharp(C#)" }, Content = "C# is intended to be a simple, modern,
general-purpose, object-oriented programming language. Its development team
is led by Anders Hejlsberg.The most recent version is C# 5.0, which was
released on August 15, 2012." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "Java" }, Content = "Java is a set of computer software and specifications
developed by Sun Microsystems, later acquired by Oracle Corporation, that
provides a system for developing application software and deploying it in a
cross - platform computing environment.Java is used in a wide variety of
computing platforms from embedded devices and mobile phones to enterprise
servers and supercomputers.While less common, Java applets run in secure,
sandboxed environments to provide many features of native applications and
can be embedded in HTML pages." });
 adaptiveItems.Add(new TabTabItem { Header = new TabHeader { Text
= "VB.Net" }, Content = "The command-line compiler, VBC.EXE, is installed as
part of the freeware .NET Framework SDK. Mono also includes a command - line
VB.NET compiler.The most recent version is VB 2012, which was released on
August 15, 2012." });
 }
 }
}
```



```

 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "Xamarin" }, Content = "Xamarin is a San Francisco, California based
software company created in May 2011[3] by the engineers that created Mono,
[4] Mono for Android and MonoTouch that are cross - platform implementations
of the Common Language Infrastructure(CLI) and Common Language
Specifications(often called Microsoft.NET).With a C#-shared codebase,
developers can use Xamarin tools to write native Android, iOS, and Windows
apps with native user interfaces and share code across multiple
platforms.[5] Xamarin has over 1 million developers in more than 120
countries around the World as of May 2015." });
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "ASP.NET" }, Content = "ASP.NET is an open-source server-side web
application framework designed for web development to produce dynamic web
pages.It was developed by Microsoft to allow programmers to build dynamic
web sites, web applications and web services.It was first released in
January 2002 with version 1.0 of the.NET Framework, and is the successor to
Microsoft Active Server Pages (ASP) technology. ASP.NET is built on the
Common Language Runtime (CLR), allowing programmers to write ASP.NET code
using any supported .NET language. The ASP.NET SOAP extension framework
allows ASP.NET components to process SOAP messages." });
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "ASP.NET MVC" }, Content = "The ASP.NET MVC is a web application framework
developed by Microsoft, which implements the model-view-controller(MVC)
pattern.It is open - source software, apart from the ASP.NET Web Forms
component which is proprietary.In the later versions of ASP.NET, ASP.NET
MVC, ASP.NET Web API, and ASP.NET Web Pages(a platform using only Razor
pages) will merge into a unified MVC 6.The project is called ASP.NET vNext."
});
 adaptiveItems.Add(new TabBarItem { Header = new TabHeader { Text
= "JavaScript" }, Content = "JavaScript (JS) is an interpreted computer
programming language. It was originally implemented as part of web browsers
so that client - side scripts could interact with the user, control the
browser, communicate asynchronously, and alter the document content that was
displayed.[5] More recently, however, it has become common in both game
development and the creation of desktop applications." });
 ViewBag.adaptiveItems = adaptiveItems;
 ViewBag.dropDownData = dropdownDatasource();
 return View();
 }
 public List<dropdownData> dropdownDatasource()
 {
 List<dropdownData> dropdownDatas = new List<dropdownData>();
 dropdownDatas.Add(new dropdownData { Id = 0, Text= "HTML" });
 dropdownDatas.Add(new dropdownData { Id = 1, Text = "C
Sharp (C#)" });
 dropdownDatas.Add(new dropdownData { Id = 2, Text = "VB.Net" });
 dropdownDatas.Add(new dropdownData { Id = 3, Text = "Xamarin"
});
 dropdownDatas.Add(new dropdownData { Id = 4, Text = "ASP.NET"
});
 dropdownDatas.Add(new dropdownData { Id = 5, Text = "ASP.NETMVC"
});
 dropdownDatas.Add(new dropdownData { Id = 6, Text = "JavaScript"
});
 return dropdownDatas;
 }
 public class dropdownData

```

```

 {
 public string Text { get; set; }
 public int Id { get; set; }
 }
}

```

### Enabling tab key navigation in Tabs

The **TabIndex** property of a Tab item is used to enable tab key navigation for that particular item. When a positive value is assigned to the **TabIndex** property, it allows the user to switch focus to the next or previous tab item using the Tab or Shift+Tab keys. By default, the user can only switch between tab items using the arrow keys.

If the **TabIndex** value is set to 0 for all tab items, the tab will switch based on the order of the elements on the page. This means that if the tab items are listed in a specific order on the page, the user will be able to navigate through them using the Tab key in that same order.

To use the **TabIndex** property, you can assign a positive value to the property of each tab item that you want to enable tab key navigation. For example:

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Tab("ej2Tab")
 .Width("500px")
 .Items(new List<TabBarItem> {
 new TabBarItem { Header = ViewBag.headerText0, Content = "HyperText Markup Language, commonly referred to as HTML, is the standard markup language used to create web pages. Along with CSS, and JavaScript, HTML is a cornerstone technology, used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.[1] Web browsers can read HTML files and render them into visible or audible web pages. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language, rather than a programming language.", TabIndex = 0 },
 new TabBarItem { Header = ViewBag.headerText1, Content = "C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 5.0, which was released on August 15, 2012.", TabIndex = 0 },
 new TabBarItem { Header = ViewBag.headerText2, Content = "Java is a set of computer software and specifications developed by Sun Microsystems, later acquired by Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. While less common, Java applets run in secure, sandboxed environments to provide many features of native applications and can be embedded in HTML pages.", TabIndex = 0 },
 new TabBarItem { Header = ViewBag.headerText3, Content = "The command-line compiler, VBC.EXE, is installed as part of the freeware .NET Framework SDK. Mono also includes a command-line VB.NET compiler. The most recent version is VB 2012, which was released on August 15, 2012.", TabIndex = 0 },
 })

```

```

 new TabBarItem { Header = ViewBag.headerText4, Content = "Xamarin is
a San Francisco, California based software company created in May 2011[3] by
the engineers that created Mono,[4] Mono for Android and MonoTouch that are
cross-platform implementations of the Common Language Infrastructure (CLI)
and Common Language Specifications (often called Microsoft .NET). With a C#-
shared codebase, developers can use Xamarin tools to write native Android,
iOS, and Windows apps with native user interfaces and share code across
multiple platforms.[5] Xamarin has over 1 million developers in more than
120 countries around the World as of May 2015.", TabIndex = 0 },
 new TabBarItem { Header = ViewBag.headerText5, Content = "ASP.NET is
an open-source server-side web application framework designed for web
development to produce dynamic web pages. It was developed by Microsoft to
allow programmers to build dynamic web sites, web applications and web
services. It was first released in January 2002 with version 1.0 of the .NET
Framework, and is the successor to Microsoft's Active Server Pages (ASP)
technology. ASP.NET is built on the Common Language Runtime (CLR), allowing
programmers to write ASP.NET code using any supported .NET language. The
ASP.NET SOAP extension framework allows ASP.NET components to process SOAP
messages.", TabIndex = 0 },
 new TabBarItem { Header = ViewBag.headerText6, Content = "The
ASP.NET MVC is a web application framework developed by Microsoft, which
implements the model-view-controller (MVC) pattern. It is open-source
software, apart from the ASP.NET Web Forms component which is proprietary.
In the later versions of ASP.NET, ASP.NET MVC, ASP.NET Web API, and ASP.NET
Web Pages (a platform using only Razor pages) will merge into a unified MVC
6.The project is called ASP.NET vNext.", TabIndex = 0 },
 new TabBarItem { Header = ViewBag.headerText7, Content = "JavaScript
(JS) is an interpreted computer programming language. It was originally
implemented as part of web browsers so that client-side scripts could
interact with the user, control the browser, communicate asynchronously, and
alter the document content that was displayed.[5] More recently, however, it
has become common in both game development and the creation of desktop
applications.", TabIndex = 0 }
 })
 .HeightAdjustMode(HeightStyles.Auto)
 .OverflowMode(OverflowMode.Scrollable)
 .Render()
)
<style>
 #container {
 visibility: hidden;
 max-width: 650px;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 .e-content .e-item {
 font-size: 12px;
 margin: 10px;
 text-align: justify;
 }
</style>

```

## TABKEYNAVIGATION.CS

```
public ActionResult Index()
{
 ViewBag.headerText0 = new TabHeader { Text = "HTML" };
 ViewBag.headerText1 = new TabHeader { Text = "C Sharp (C#)" };
 ViewBag.headerText2 = new TabHeader { Text = "Java" };
 ViewBag.headerText3 = new TabHeader { Text = "VB.Net" };
 ViewBag.headerText4 = new TabHeader { Text = "Xamarin" };
 ViewBag.headerText5 = new TabHeader { Text = "ASP.NET" };
 ViewBag.headerText6 = new TabHeader { Text = "ASP.NET MVC" };
 ViewBag.headerText7 = new TabHeader { Text = "JavaScript" };
 return View();
}
```

With this code, the user will be able to switch between the tab items using the Tab and Shift+Tab keys, in the order specified by the `TabIndex` values.

It's important to note that the `TabIndex` property only affects the ability to navigate between tab items using the Tab key. The user will still be able to use the arrow keys to switch between tab items, regardless of the value of the `TabIndex` property.

## Migration from Essential JS 1

This article describes the API migration process of Tab component from Essential JS 1 to Essential JS 2.

### Accessibility

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Keyboard Navigation | **Property:** `AllowKeyboardNavigation` <br /><br />

@Html.EJ().Tab("ejTab").AllowKeyboardNavigation(true).Render() <br /> | **Not Applicable** |

| Localization | **Not Applicable** | **Property:** `Locale` <br /><br /> @Html.EJS().Tab("ej2Tab").Locale("en-US").Render() <br /> |

| RTL | **Property:** `EnableRTL` <br /><br /> @Html.EJ().Tab("ejTab").EnableRTL(true).Render() <br /> |

**Property:** `EnableRtl` <br /><br /> @Html.EJS().Tab("ej2Tab").EnableRtl(true).Render() <br /> |

### AjaxSettings

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Default | **Property:** `AjaxSettings` <br /><br /> @Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br /> &#160; ajax.Type("GET"); <br /> }).Render() <br /> | **Not Applicable** |

| Asynchronous | **Property:** `AjaxSettings.Async` <br /><br /> @Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br /> &#160; ajax.Async(true); <br /> }).Render() <br /> | **Not Applicable** |

| Browser Cache | **Property:** `AjaxSettings.Cache` <br /><br /> @Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br /> &#160; ajax.Cache(false); <br /> }).Render() <br /> | **Not Applicable** |

| Request type | **Property:** *AjaxSettings.ContentType* <br /><br />  
@Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br /> &#160; ajax.ContentType("html"); <br />  
}).Render() <br /> | **Not Applicable** |

| Data | **Property:** *AjaxSettings.Data* <br /><br /> @Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br />  
&#160; ajax.Data(""); <br /> }).Render() <br /> | **Not Applicable** |

| Response type | **Property:** *AjaxSettings.DataType* <br /><br />  
/>@Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br /> &#160; ajax.DataType("html"); <br />  
}).Render() <br /> | **Not Applicable** |

| HTTP request type | **Property:** *AjaxSettings.Type* <br /><br />  
@Html.EJ().Tab("ejTab").AjaxSettings(ajax => { <br /> &#160; ajax.Type("GET"); <br /> }).Render() <br />  
/> | **Not Applicable** |

| AjaxBeforeLoad | **Event:** *AjaxBeforeLoad* <br /><br /> @Html.EJ().Tab("ejTab").ClientSideEvents(e =>  
e.AjaxBeforeLoad("onAjaxBeforeLoad")).Render() <br /> | **Not Applicable** |

| AjaxError | **Event:** *AjaxError* <br /><br /> @Html.EJ().Tab("ejTab").ClientSideEvents(e =>  
e.AjaxError("onAjaxError")).Render() <br /> | **Not Applicable** |

| AjaxLoad | **Event:** *AjaxLoad* <br /><br /> @Html.EJ().Tab("ejTab").ClientSideEvents(e =>  
e.AjaxLoad("onAjaxLoad")).Render() <br /> | **Not Applicable** |

| AjaxSuccess | **Event:** *AjaxSuccess* <br /><br /> @Html.EJ().Tab("ejTab").ClientSideEvents(e =>  
e.AjaxSuccess("onAjaxSuccess")).Render() <br /> | **Not Applicable** |

## Animation

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Not Applicable** | **Property:** *Animation* <br /><br />  
@Html.EJS().Tab("ej2Tab").Animation(anim => { <br /> &#160; anim.Previous("").Next(""); <br />  
}).Render() <br /> |

| EnableAnimation | **Property:** *EnableAnimation* <br /><br />  
@Html.EJ().Tab("ejTab").EnableAnimation(false).Render() <br /> | **Not Applicable** |

| Previous animation | **Not Applicable** | **Property:** *Animation.Prev* <br /><br />  
@Html.EJS().Tab("ej2Tab").Animation(anim => { <br /> &#160; anim.Previous(new  
List<TabTabActionSettings>() { <br /> &#160; &#160; new TabTabActionSettings() { Effect =  
"SlideRight" } <br /> &#160; }); <br /> }).Render() <br /> |

| Next animation | **Not Applicable** | **Property:** *Animation.Next* <br /><br />  
@Html.EJS().Tab("ej2Tab").Animation(anim => { <br /> &#160; anim.Next(new  
List<TabTabActionSettings>() { <br /> &#160; &#160; new TabTabActionSettings() { Effect = "SlideLeft"  
} <br /> &#160; }); <br /> }).Render() <br /> |

| Duration <br /> [prev / next] | **Not Applicable** | **Property:** *Animation.Next.Duration* <br /><br />  
@Html.EJS().Tab("ej2Tab").Animation(anim => { <br /> &#160; anim.Next(new  
List<TabTabActionSettings>() { <br /> &#160; &#160; new TabTabActionSettings() { Duration = 400 }  
<br /> &#160; }); <br /> }).Render() <br /> |

| Easing <br /> [prev / next] | **Not Applicable** | **Property:** *Animation.Next.Easing* <br /><br />  
@Html.EJS().Tab("ej2Tab").Animation(anim => { <br /> &#160; anim.Next(new

List<TabTabActionSettings>() { <br /> &#160; &#160; new TabTabActionSettings() { Easing = "ease-in" } <br /> &#160; }; <br /> }.Render() <br /> |

| Effect <br /> [prev / next] | **Not Applicable** | **Property:** *Animation.Next.Effect* <br /><br /> @Html.EJS().Tab("ej2Tab").Animation(anim => { <br /> &#160; anim.Next(new List<TabTabActionSettings>() { <br /> &#160; &#160; new TabTabActionSettings() { Effect = "SlideRight" } <br /> &#160; }; <br /> }).Render() <br /> |

## Header

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Header position | **Property:** *HeaderPosition* <br /><br />

@Html.EJS().Tab("ej2Tab").HeaderPosition(HeaderPosition.Top).Render() <br /> | **Property:** *HeaderPlacement* <br /><br />

@Html.EJS().Tab("ej2Tab").HeaderPlacement(Syncfusion.EJ2.Navigations.HeaderPosition.Bottom).Render() <br /> |

| Header size | **Property:** *HeaderSize* <br /><br /> @Html.EJS().Tab("ej2Tab").HeaderSize("50px").Render() <br /> | **Not Applicable** |

| OverflowModes | **Not Applicable** | **Property:** *OverflowMode* <br /><br />

@Html.EJS().Tab("ej2Tab").OverflowMode(Syncfusion.EJ2.Navigations.OverflowMode.Popup).Render() <br /> |

| TabScroll | **Property:** *EnableTabScroll* <br /><br />

@Html.EJS().Tab("ej2Tab").EnableTabScroll(false).Render() <br /> | **Not Applicable** |

## Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Default | **Not Applicable** | **Property:** *Items* <br /><br /> @Html.EJS().Tab("ej2Tab").Items(item => { <br /> &#160; item.Header(h => { h.Text("Tab1"); }).Content("contents").Add(); <br /> }).Render() <br /> |

| Content | **Not Applicable** | **Property:** *Items[0].Content* <br /><br />

@Html.EJS().Tab("ej2Tab").Items(item => { <br /> &#160; item.Content("contents").Add(); <br /> }).Render() <br /> |

| Custom Class | **Not Applicable** | **Property:** *Items[0].CssClass* <br /><br />

@Html.EJS().Tab("ej2Tab").Items(item => { <br /> &#160; item.CssClass("customClass").Add(); <br /> }).Render() <br /> |

| Header | **Not Applicable** | **Property:** *Items[0].Header* <br /><br />

@Html.EJS().Tab("ej2Tab").Items(item => { <br /> &#160; item.Header(h => { h.Text("Tab1"); }).Add(); <br /> }).Render() <br /> |

| Icon class | **Not Applicable** | **Property:** *Items[0].Header.IconCss* <br /><br />

@Html.EJS().Tab("ej2Tab").Items(item => { <br /> &#160; item.Header(h => { h.IconCss("e-icon"); }).Add(); <br /> }).Render() <br /> |

```
| Icon position | Not Applicable | Property: Items[0].Header.IconPosition

@Html.EJS().Tab("ej2Tab").Items(item => {
 item.Header(h => { h.IconPosition("Left");
}).Add();
 }).Render()
 |

| Header text | Not Applicable | Property: Items[0].Header.Text

@Html.EJS().Tab("ej2Tab").Items(item => {
 item.Header(h => { h.Text("Tab1"); }).Add();

 }).Render()
 |

| Get items length | Method: getItemsCount()

 var tabObj = $("#tab").data("ejTab");

tabObj.getItemsCount();
 | Not Applicable |

| Add Items | Method: addItem(url, displayLabel, index, cssClass, id)

 var tabObj =
$("#tab").data("ejTab");
 tabObj.addItem("#new", "New Item", 3, "myClass", "newItem");
 |
Method: addTab(items, index)

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.addTab([{
 header: { text:
'Tab1' },
 content: 'contents' }], 1
);
 |

| BeforeAdd | Not Applicable | Event: Adding

@Html.EJS().Tab("ej2Tab").Adding("onAdding").Render()
 |

| AfterAdd | Event: ItemAdd

 @Html.EJ().Tab("ejTab").ClientSideEvents(e =>
e.ItemAdd("itemAdd")).Render()
 | Event: Added

@Html.EJS().Tab("ej2Tab").Added("onAdded").Render()
 |

| Remove item | Method: removeItem(index)

 var tabObj = $("#tab").data("ejTab");

tabObj.removeItem(1);
 | Method: removeTab(index)

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.removeTab(1);
 |

| BeforeRemove | Event: BeforeItemRemove

 @Html.EJ().Tab("ejTab").ClientSideEvents(e
=> e.BeforeItemRemove("beforeItemRemove")).Render()
);
 | Event: Removing

@Html.EJS().Tab("ej2Tab").Removing("onRemoving").Render()
 |

| AfterRemove | Event: ItemRemove

 @Html.EJ().Tab("ejTab").ClientSideEvents(e =>
e.ItemRemove("itemRemove")).Render()
 | Event: Removed

@Html.EJS().Tab("ej2Tab").Removed("onRemoved").Render()
 |

| SelectedItemIndex | Property: SelectedItemIndex

@Html.EJ().Tab("ejTab").SelectedItemIndex("1").Render()
 | Property: SelectedItem

@Html.EJS().Tab("ej2Tab").SelectedItem("0").Render()
 |

| Select item | Not Applicable | Method: select(index)

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.select(1);
 |

| BeforeActive | Event: BeforeActive

 @Html.EJ().Tab("ejTab").ClientSideEvents(e =>
e.BeforeActive("beforeActive")).Render()
 | Event: Selecting

@Html.EJS().Tab("ej2Tab").Selecting("onSelecting").Render()
 |

| AfterActive | Event: ItemActive

 @Html.EJ().Tab("ejTab").ClientSideEvents(e =>
e.ItemActive("itemActive")).Render()
 | Event: Selected

@Html.EJS().Tab("ej2Tab").Selected("onSelected").Render()
 |

| Disable items | Property: DisabledItemIndex

@Html.EJ().Tab("ejTab").DisabledItemIndex(new List<int>() { 0, 1 }).Render()
 | Not Applicable |
```

| Enable items | **Property:** *EnabledItemIndex* <br /><br />  
@Html.EJ().Tab("ejTab").EnabledItemIndex(new List<int>() { 0, 1 }).Render() <br /> | **Not Applicable** |

| Enable/Disable item | **Not Applicable** | **Property:** *Items[0].Disabled* <br /><br />  
@Html.EJS().Tab("ej2Tab").Items(item => { <br /> &#160; item.Disabled(true).Add(); <br /> }).Render()  
<br /> |

| Hide items | **Property:** *HiddenItemIndex* <br /><br /> @Html.EJ().Tab("ejTab").HiddenItemIndex(new  
List<int>() { 0, 1 }).Render() <br /> | **Not Applicable** |

| Hide item | **Method:** *hideItem(index)* <br /><br /> var tabObj = \$("#tab").data("ejTab"); <br />  
tabObj.hideItem(1); <br /> | **Method:** *hideTab(index, true)* <br /><br /> var tab =  
document.getElementById('ej2Tab').ej2\_instances[0]; <br /> tab.hideTab(1, true); <br /> |

| Show item | **Method:** *showItem(index)* <br /><br /> var tabObj = \$("#tab").data("ejTab"); <br />  
tabObj.showItem(1); <br /> | **Method:** *hideTab(index, false)* <br /><br /> var tab =  
document.getElementById('ej2Tab').ej2\_instances[0]; <br /> tab.hideTab(1, false); <br /> |

### Common

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Collapse active item | **Property:** *Collapsible* <br /><br />  
@Html.EJ().Tab("ejTab").Collapsible(true).Render() <br /> | **Not Applicable** |

| Custom class | **Property:** *CssClass* <br /><br />  
@Html.EJ().Tab("ejTab").CssClass("cutomClass").Render() <br /> | **Property:** *CssClass* <br /><br />  
@Html.EJS().Tab("ej2Tab").CssClass("cutomClass").Render() <br /> |

| Enabled | **Property:** *Enabled* <br /><br /> @Html.EJ().Tab("ejTab").Enabled(true).Render() <br /> |  
**Method:** *disable(false)* <br /><br /> var tab = document.getElementById('ej2Tab').ej2\_instances[0]; <br />  
/> tab.disable(false); <br /> |

| Persistence | **Property:** *EnablePersistence* <br /><br />  
@Html.EJ().Tab("ejTab").EnablePersistence(true).Render() <br /> | **Property:** *EnablePersistence* <br />  
/><br /> @Html.EJS().Tab("ej2Tab").EnablePersistence(true).Render() <br /> |

| Events | **Property:** *Events* <br /><br /> @Html.EJ().Tab("ejTab").Events("click").Render() <br /> | **Not  
Applicable** |

| Height | **Property:** *Height* <br /><br /> @Html.EJ().Tab("ejTab").Height("500px").Render() <br /> |  
**Property:** *Height* <br /><br /> @Html.EJS().Tab("ej2Tab").Height("500px").Render() <br /> |

| HeightAdjustMode | **Property:** *HeightAdjustMode* <br /><br />  
@Html.EJ().Tab("ejTab").HeightAdjustMode(HeightAdjustMode.Fill).Render() <br /> | **Property:**  
*HeightAdjustMode* <br /><br />  
@Html.EJS().Tab("ej2Tab").HeightAdjustMode(Syncfusion.EJ2.Navigations.HeightStyles.Fill).Render() <br />  
/> |

| HTML Attributes | **Property:** *HtmlAttributes* <br /><br />  
@Html.EJ().Tab("ejTab").HtmlAttributes("").Render() <br /> | **Not Applicable** |

| ID prefix | **Property:** *IdPrefix* <br /><br /> @Html.EJ().Tab("ejTab").IdPrefix("e-tab").Render() <br /> |  
**Not Applicable** |



```
| ShowCloseButton | Property: ShowCloseButton

@Html.EJ().Tab("ejTab").ShowCloseButton(true).Render()
 | Property: ShowCloseButton

/> @Html.EJS().Tab("ej2Tab").ShowCloseButton(true).Render()
 |

| ShowReloadIcon | Property: ShowReloadIcon

@Html.EJ().Tab("ejTab").ShowReloadIcon(true).Render()
 | Not Applicable |

| ShowRoundedCorner | Property: ShowRoundedCorner

@Html.EJ().Tab("ejTab").ShowRoundedCorner(true).Render()
 | Not Applicable |

| Width | Property: Width @Html.EJ().Tab("ejTab").Width("500px").Render() |
Property: Width

 @Html.EJS().Tab("ej2Tab").Width("500px").Render()
 |

| Destroy | Not Applicable | Method: destroy()

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.destroy();
 |

| Disable Tab | Method: disable()

 var tabObj = $("#tab").data("ejTab");

tabObj.disable();
 | Method: disable(true)

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.disable(true);
 |

| Enable Tab | Method: enable()

 var tabObj = $("#tab").data("ejTab");

tabObj.enable();
 | Method: disable(false)

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.disable(false);
 |

| Hide Tab | Method: hide()

 var tabObj = $("#tab").data("ejTab");
 tabObj.hide();

/> | Not Applicable |

| Show Tab | Method: show()

 var tabObj = $("#tab").data("ejTab");
 tabObj.show();

 | Not Applicable |

| Refresh | Not Applicable | Method: refresh()

 var tab =
document.getElementById('ej2Tab').ej2_instances[0];
 tab.refresh();
 |

| Created | Event: Create

 @Html.EJ().Tab("ejTab").ClientSideEvents(e =>
e.Create("onCreate")).Render()
 | Event: Created

@Html.EJS().Tab("ej2Tab").Created("onCreated").Render()
 |

| Destroyed | Event: Destroy

 @Html.EJ().Tab("ejTab").ClientSideEvents(e =>
e.Destroy("onDestroy")).Render()
 | Event: Destroyed

@Html.EJS().Tab("ej2Tab").Destroyed("onDestroyed").Render()
 |
```

## TextBox

### Getting Started with ASP.NET MVC TextBox Control

This section briefly explains about how to include [ASP.NET MVC TextBox](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

#### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

## ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC TextBox control

Now, add the Syncfusion ASP.NET MVC TextBox control in ~/Views/Home/Index.cshtml page.

## CSHTML

```
<div class="control-section">
 <div class="control_wrapper textbox-control-section">
 @Html.EJS().TextBox("firstname").Placeholder("First Name").Render()
 </div>
</div>
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC TextBox control will be rendered in the default web browser.

First Name

---

### Adding icons to the TextBox

You can create a TextBox with icon as a group by creating the parent div element with the class `e-input-group` and add the icon element as span with the class `e-input-group-icon`. For detailed information, refer to the [Groups](#) section.

## CSHTML

```
<div class="control-section">
 <div class="control_wrapper textbox-control-section">
 <div class="e-input-group">
 <input class="e-input" name='input' type="text"
placeholder="Enter Date" />

 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 var inputObject = {};
 var input = document.querySelectorAll('.e-input-group .e-input, .e-float-
input.e-input-group input');
 var inputIcon = document.querySelectorAll('.e-input-group-icon');
 var focusFn = function (index) {
 return function () {
 getParentNode(input[index]).classList.add('e-input-focus');
 };
 };
 var blurFn = function (index) {
```

```

 return function () {
getParentNode(input[index]).classList.remove('e-input-focus'); };
 };
 var mouseupFn = function () {
 var ele = this; setTimeout(function () {
 ele.classList.remove('e-input-btn-ripple');
 }, 500);
 };
 for (var i = 0; i < input.length; i++) {
 input[i].addEventListener('focus', focusFn(i));
 input[i].addEventListener('blur', blurFn(i));
 }
 for (var j = 0; j < inputIcon.length; j++) {
 inputIcon[j].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[j].addEventListener('mouseup', mouseupFn);
 }
 function getParentNode(element) {
 var parentNode = element.parentNode;
 if (parentNode.classList.contains('e-input-in-wrap')) {
 return parentNode.parentNode;
 }
 return parentNode;
 }
</script>
<style>
 .e-input-group-icon:before {
 font-family: e-icons;
 }
 .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
 font-size: 16px;
 }
 .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
 content: "□";
 }
 .e-input-group-icon.e-input-up:before { /* csslint allow: adjoining-
classes */
 content: '\e85e';
 }
</style>

```

### Floating label

The floating label TextBox floats the label above the TextBox after focusing, or filled with value in the TextBox. You can create the floating label TextBox by using the [FloatLabelType](#) property.

### CSHTML

```

<div class="control-section">
 <div class="control_wrapper textbox-control-section">
 @Html.EJS().TextBox("firstname").Placeholder("First
Name").FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabelType.Auto).Render()
 </div>
</div>

```

First Name

**Note:** [View Sample in GitHub.](#)

See also

- [How to render TextBox programmatically](#)
- [How to add floating label to TextBox programmatically](#)

## Groups

The following section explains you the steps required to create TextBox with **icon** and **floating label**.

### TextBox:

- Create a parent div element with the class **e-input-group**
- Place input element with the class **e-input** inside the parent div element.

```
`html
```

```
<div class="e-input-group">
```

```
<input class="e-input" name='input' type="text" placeholder="Enter Date"/>
```

```
</div>
```

```
,
```

### Floating label:

- Add the **e-float-input** class to the parent div element.
- Remove the **e-input** class and add **required** attribute to the input element.
- Place the span element with class **e-float-line** after the input element.
- Place the label element with class **e-float-text** after the above created span element. When you focus or filled with value in the TextBox, the label floats above the TextBox.

**Note:** Creating the Floating label TextBox, you have to set the **required** attribute to the Input element to achieve the floating label functionality which is used for validating the value existence in TextBox. If you want to render the Floating label TextBox without **required** attribute then refer to the [Floating label without required attribute](#) section.

```
`html
```

```
<div class="e-float-input e-input-group">
```

```
<input type="text" required/>
```

```

```

```
<label class="e-float-text">Enter Name </label>
```

</div>

And refer to the following sections to add the icons to the TextBox.

#### With icon and floating label

Create an icon element as a span with the class `e-input-group-icon`, and the user can place the icon in either side of TextBox by adding the created icon element before/after the input.

For the floating label enabled TextBox add the icon element as first or last element inside the TextBox wrapper, and based on the element position it will act as prefix or suffix icon.

#### CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <h4> TextBox with icons </h4>
 <div class="e-input-group">
 <input class="e-input" type="text" placeholder="Enter Date" />

 </div>
 <div class="e-input-group e-float-icon-left">

 <div class="e-input-in-wrap">
 <input class="e-input" type="text" placeholder="Enter Date"
 />
 </div>
 </div>
 <div class="e-input-group e-float-icon-left">

 <div class="e-input-in-wrap">
 <input class="e-input" type="text" placeholder="Enter Date"
 />

 </div>
 </div>
 <h4> Floating label with icons </h4>
 <div class="e-float-input e-input-group">
 <input required type="text" />

 <label class="e-float-text"> Enter Date </label>

 </div>
 <div class="e-float-input e-input-group e-float-icon-left">

 <div class="e-input-in-wrap">
 <input required type="text" />

 <label class="e-float-text"> Enter Date </label>
 </div>
 </div>
 <div class="e-float-input e-input-group e-float-icon-left">

 <div class="e-input-in-wrap">
 <input required type="text" />

 <label class="e-float-text"> Enter Date </label>
 </div>
 </div>
 </div>
</div>
```

```

</div>
</div>
</div>
</div>
<script>
 ej.base.enableRipple(true);
 var inputObject = {};
 var input = document.querySelectorAll('.e-input-group .e-input,.e-float-
input.e-input-group input');
 var inputIcon = document.querySelectorAll('.e-input-group-icon');
 var focusFn = function (index) {
 return function () {
 parentNode(input[index]).classList.add('e-input-focus');
 };
 };
 var blurFn = function (index) {
 return function () {
 parentNode(input[index]).classList.remove('e-input-focus');
 };
 };
 var mouseupFn = function () {
 var ele = this; setTimeout(function () {
 ele.classList.remove('e-input-btn-ripple');
 }, 500);
 };
 for (var i = 0; i < input.length; i++) {
 input[i].addEventListener('focus', focusFn(i));
 input[i].addEventListener('blur', blurFn(i));
 }
 for (var j = 0; j < inputIcon.length; j++) {
 inputIcon[j].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[j].addEventListener('mouseup', mouseupFn);
 }
 function getParentNode(element) {
 var parentNode = element.parentNode;
 if (parentNode.classList.contains('e-input-in-wrap')) {
 return parentNode.parentNode;
 }
 return parentNode;
 }
}
</script>
<style>
.e-input-group-icon:before {
 font-family: e-icons;
}
.e-input-group .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
 font-size: 16px;
}
.e-input-group.e-small .e-input-group-icon.e-input-popup-date { /*
csslint allow: adjoining-classes */
 font-size: 14px;
}
.e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
```

```

 content: "□";
 }
 .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
 content: "□";
 }
 .e-input-group-icon.e-input-date:before { /* csslint allow: adjoining-
classes */
 content: "□";
 }
}
</style>

```

### ICON.CS

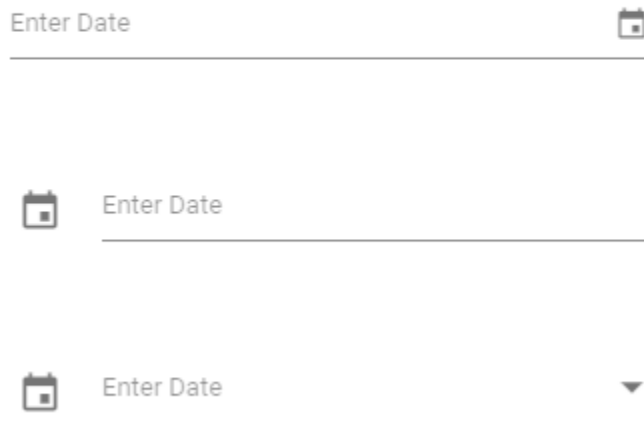
```

public ActionResult Index()
{
 return View();
}

```

Output be like the below.

### TextBox with icons



With clear button and floating label

The clear button is added to the input for clearing the value given in the TextBox.

It is shown only when the input field has a value, otherwise not shown.

You can add the clear button to the TextBox by enabling `showClearButton` API.

### CSHTML

```

<div class="control-section">
 <div class="control_wrapper accordion-control-section">

```



```
<h4> TextBox with clear button </h4>
@Html.EJS().TextBox("firstname").Placeholder("First
Name").ShowClearButton(true).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabel
Type.Never).Render()
<h4> Floating TextBox with clear button </h4>
@Html.EJS().TextBox("lastname").Placeholder("Last
Name").ShowClearButton(true).FloatLabelType(Syncfusion.EJ2.Inputs.FloatLabel
Type.Auto).Render()
</div>
</div>
```

### CLEAR.CS

```
public ActionResult Index()
{
 return View();
}
```

Output be like the below.

#### TextBox with clear button

John ×

#### Floating TextBox with clear button

Enter Name  
John ×

### Floating label without required attribute

You can render the Floating label TextBox without required attribute by manually float the label above of the TextBox using input events. You can manually float the label above of the TextBox by adding the below list of classes to the floating label element. The classes are:

Class Name | Description

e-label-top | Floats the label above of the TextBox.

e-label-bottom | Label to be placed as placeholder for the TextBox.

### CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <div class="e-float-input">
 <input type="text" id='inpt1' />

```

```

 <label class="e-float-text"> Enter Value </label>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 /* To get the Input element */
 var inputElement = document.getElementById('inpt1');
 /* Update the label position based on Input value */
 updateLabelState(inputElement.value,
inputElement.parentElement.querySelector('.e-float-text'));
 inputElement.addEventListener("focus", function () {
 var label = this.parentElement.querySelector('.e-float-text');
 label.classList.add('e-label-bottom');
 label.classList.remove('e-label-top');
 });
 inputElement.addEventListener("blur", function () {
 updateLabelState(this.value,
this.parentElement.querySelector('.e-float-text'));
 });
 inputElement.addEventListener("input", function () {
 updateLabelState(this.value,
this.parentElement.querySelector('.e-float-text'));
 });
 /* Update the label position based on Input value */
 /* e-label-top - Floats the label above of the TextBox */
 /* e-label-bottom - Label to be placed as placeholder for the
TextBox */
 function updateLabelState(value, label) {
 if (value) {
 label.classList.add('e-label-top');
 label.classList.remove('e-label-bottom');
 } else {
 label.classList.add('e-label-bottom');
 label.classList.remove('e-label-top');
 }
 }
</script>
<style>
 .control-section {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 260px;
 }
</style>

```

### FLOATING.CS

```

public ActionResult Index()
{
 return View();
}

```

### Multi-line input with floating label

Add the HTML `textarea` element with the `e-input` class to create default multi-line input.

Add the floating label support to the `multi-line input` by creating the floating label structure as defined in the initial section.

#### CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <textarea class="e-input"> Address </textarea>
 <div class="e-float-input">
 <textarea required></textarea>

 <label class="e-float-text"> Address </label>
 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect
when focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener('focus', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener('blur', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-
focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
 }
</script>
```

#### MULTILINE.CS

```
public ActionResult Index()
{
 return View();
}
```

## See Also

- [How to add floating label to TextBox programmatically](#)
- [Change the floating label color of the TextBox](#)
- [Change the color of the TextBox based on its value](#)

## Sizing

You can render the TextBox in two different sizes.

Property | Description

Normal | By default, the TextBox is rendered with normal size.

Small | You need to add `e-small` class to the input element, or else add to the input container.

## CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <h4> Normal Size </h4>
 <div class="e-input-group">
 <input class="e-input" type="text" placeholder="Enter Name" />
 </div>
 <div class="e-float-input">
 <input type='text' required />

 <label class="e-float-text">Enter Name</label>
 </div>
 <div class="e-input-group">
 <input class="e-input" type="text" placeholder="Enter Date" />

 </div>
 <h4> Small Size </h4>
 <div class="e-input-group e-small">
 <input class="e-input" type="text" placeholder="Enter Name" />
 </div>
 <div class="e-float-input e-small">
 <input type='text' required />

 <label class="e-float-text">Enter Name</label>
 </div>
 <div class="e-input-group e-small">
 <input class="e-input" type="text" placeholder="Enter Date" />

 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
 focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
```

```

 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener("blur", function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (var i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[i].addEventListener('mouseup', function () {
 var element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
}
</script>
<style>
 .e-input-group-icon:before {
 font-family: e-icons;
 }
 .e-input-group .e-input-group-icon.e-input-popup-date { /* csslint
allow: adjoining-classes */
 font-size: 16px;
 }
 .e-input-group.e-small .e-input-group-icon.e-input-popup-date { /*
csslint allow: adjoining-classes */
 font-size: 14px;
 }
 .e-input-group-icon.e-input-popup-date:before { /* csslint allow:
adjoining-classes */
 content: "□";
 }
</style>

```

### SIZING.CS

```

public ActionResult Index()
{
 return View();
}

```

## Validation

The TextBox supports three types of validation styles namely error, warning, and success. These states are enabled by adding corresponding classes `.e-error`, `.e-warning`, or `.e-success` to the input parent element.

### CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <div class="e-input-group e-warning">
 <input class="e-input" type="text" placeholder="Input with
warning" />
 </div>
 <div class="e-input-group e-error">
 <input class="e-input" type="text" placeholder="Input with
error" />
 </div>
 <div class="e-input-group e-success">
 <input class="e-input" type="text" placeholder="Input with
success" />
 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener("blur", function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
 }
 // Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
 var inputIcon = document.querySelectorAll('.e-input-group-icon');
 for (var i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[i].addEventListener('mouseup', function () {
```

```

 var element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
}
</script>

```

### VALIDATION.CS

```

public ActionResult Index()
{
 return View();
}

```

Output be like the below.

Input with warning



Input with error



Input with success



### Multiline TextBox

This feature allows the textbox to accept one or more lines of text like address, description, comments, and more.

#### Create multiline textbox

You can convert the default textbox into the multiline textbox by setting the [multiline](#) API value as true or pass HTML5 textarea as element to the textbox.

**Note:** The multiline textbox allows you to resize it in vertical direction alone.

### CSHTML

```

<div class="multiline">
 @Html.EJS().TextBox("default").Multiline(true).Placeholder("Enter your
address").Value("Mr. Dodsworth Dodsworth, System Analyst, Studio
103").FloatLabelType(FloatLabelType.Auto).Render()
</div>
<style>
 .multiline{
 margin: 30px auto;
 }
</style>

```

```
width: 20%;
}
</style>
```

### TEXTAREA.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class TextBoxController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



### Implementing floating label

You can achieve the floating label behavior in the multiline textbox by setting [floatLabelType](#) as 'Auto'. The placeholder text act as floating label to the multiline textbox. You can provide the placeholder text to the multiline textbox either by using the [placeholder](#) property or placeholder attribute.

### CSHTML

```
<label>Float label type as auto</label>
<div class="multiline">
 @Html.EJS().TextBox("multiline-auto").Multiline(true).Placeholder("Enter
your address").FloatLabelType(FloatLabelType.Auto).Render()
</div>
<label>Float label type as always</label>
<div class="multiline">
 @Html.EJS().TextBox("multiline-
always").Multiline(true).Placeholder("Enter your
address").FloatLabelType(FloatLabelType.Always).Render()
</div>
<label>Float label type as never</label>
<div class="multiline">
 @Html.EJS().TextBox("multiline-
never").Multiline(true).Placeholder("Enter your
address").FloatLabelType(FloatLabelType.Never).Render()
```



```
</div>
<style>
 .multiline{
 margin: 30px auto;
 width: 20%;
 }
</style>
```

## FLOAT.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class TextBoxController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.

### Float label type auto

Enter your address

---

### Float label type always

Enter your address

---

### Float label type never

Mr. Dodsworth Dodsworth, System  
Analyst, Studio 103

---

### Auto resizing

By default, you can manually resize the multiline textbox. But you can also create an **auto resizing** multiline textbox with both the initial and dynamic value change. It can be done by calculating the height of the text area in the created event for initial value update and in the input event for dynamic value update of the auto resize multiline textbox, as explained in the following code sample.

#### CSHTML

```
<div class="multiline">
 @Html.EJS().TextBox("default").Multiline(true).Placeholder("Enter your
address").Value("Mr. Dodsworth Dodsworth, System Analyst, Studio 103, The
Business
Center").FloatLabelType(FloatLabelType.Auto).Created("createHandler").Input(
"inputHandler").Render()
</div>
<script>
 function createHandler(args) {
 var textareaObj =
document.getElementById('default').ej2_instances[0];
 textareaObj.addAttributes({ rows: 1 });
 this.respectiveElement.style.height = "auto";
 this.respectiveElement.style.height =
(this.respectiveElement.scrollHeight) + "px";
 }
 function inputHandler(args) {
 this.respectiveElement.style.height = "auto";
 this.respectiveElement.style.height =
(this.respectiveElement.scrollHeight) + "px";
 }
</script>
<style>
 .multiline {
 margin: 30px auto;
 width: 20%;
 }
</style>
```

#### AUTO-RESIZE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class TextBoxController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

### Disable resizing

By default, the multiline textbox is rendered with resizable. You can disable the resize of the multiline textbox by applying the following CSS styles.

`CSS

```

textarea.e-input,
.e-float-input textarea,
.e-float-input.e-control-wrapper textarea,
.e-input-group textarea,
.e-input-group.e-control-wrapper textarea {
resize: none;
}
`

```

### CSHTML

```

<div class="multiline">

@Html.EJS().TextBox("default").Multiline(true).FloatLabelType(FloatLabelType
.Auto).Placeholder("Enter your address").Render()
</div>
<style>
 .multiline {
 margin: 30px auto;
 width: 20%;
 }
 textarea.e-input,
 .e-float-input textarea,
 .e-float-input.e-control-wrapper textarea,
 .e-input-group textarea,
 .e-input-group.e-control-wrapper textarea{
 resize: none;
 }
</style>

```

### DISABLE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class TextBoxController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}

```

```
}

```

Output be like the below.

Enter your address

---

### Limit the text length

By default, the text length of the multiline textbox is unlimited. You can limit the text length by setting the `maxLength` attribute using the `addAttributes` method.

### CSHTML

```
<label>Add max length attribute through inline</label>
<div class="multiline">
 @Html.EJS().TextBox("length").Multiline(true).Placeholder("Enter your
address").FloatLabelType(FloatLabelType.Auto).Render()
</div>
<label>Add maxlength attribute through addAttributes method</label>
<div class="multiline">
 @Html.EJS().TextBox("attr").Multiline(true).Placeholder("Enter your
address").FloatLabelType(FloatLabelType.Auto).Render()
</div>
@Html.EJS().Button("primarybtn").Content("Add max length").Render()
<script>
 document.getElementById("primarybtn").addEventListener('click', function
 () {
 var textareaObj = document.getElementById('attr').ej2_instances[0];
 textareaObj.addAttributes({ maxlength: 15 });
 })
</script>
<style>
 .multiline {
 margin: 30px auto;
 width: 20%;
 }
</style>
```

### MAXLENGTH.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class TextBoxController : Controller
 {
 public IActionResult DefaultFunctionalities()
 }
}
```

```

 {
 return View();
 }
 }
}

```

### Count characters

You can show the number of characters entered inside the textarea by calculating the character count in the input event of multiline textbox. The character count is updated while entering or deleting any character inside the textarea. The character count shows how many characters can be entered or left to be entered.

### CSHTML

```

<div class="multiline">
 @Html.EJS().TextBox("default").Multiline(true).Placeholder("Enter your
address").Input("inputHandler").Render()
 <div id="numbercount"></div>
</div>
<script>
 function inputHandler(args) {
 let word, addresscountRem, addressCount;
 word = this.respectiveElement.value;
 addressCount = word.length;
 addresscountRem = document.getElementById('numbercount');
 addresscountRem.textContent = addressCount + "/25";
 }
</script>
<style>
 .multiline {
 margin: 30px auto;
 width: 20%;
 }
 #numbercount{
 margin-left: 190px;
 }
</style>

```

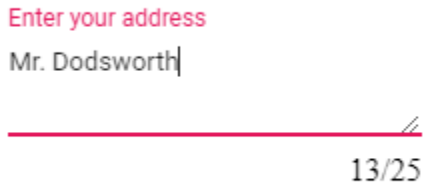
### COUNT.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class TextBoxController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}

```

Output be like the below.



### Accessibility in ASP.NET MVC Textbox component

The Textbox component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Textbox component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

```
margin: 0.5em 0;
}
```

```
</style>
```

```
<div> - All
features of the component meet the requirement.</div>
```

```
<div> - Some features of the component do not meet the requirement.</div>
```

```
<div> - The component does not meet the requirement.</div>
```

### WAI-ARIA attributes

The Textbox is characterized with complete ARIA Accessibility support that helps to access through the on-screen readers and other assistive technology devices. This component is designed with the reference of the guidelines document given in [WAI ARIA Accessibility practices](#).

The Textbox uses the `textbox` role and following ARIA properties for its element based on its state.

#### | Property | Functionality |

| --- | --- |

| `aria-placeholder` | The `aria-placeholder` is a short hint to help the users with data entry when the Textbox has no value. |

| `aria-labelledby` | The `aria-labelledby` property indicates the floating label element of the Textbox. |

### Ensuring accessibility

The Textbox component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Textbox component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Textbox component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Style and appearance in TextBox Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the appearance of TextBox wrapper element

Use the following CSS to customize the appearance of wrapper element.

```
`css
```

```
/ To specify height and font size /
```

```
.e-input:not(:valid), .e-input:valid, .e-float-input.e-control-wrapper input:not(:valid), .e-float-input.e-
control-wrapper input:valid, .e-float-input input:not(:valid), .e-float-input input:valid, .e-input-group
input:not(:valid), .e-input-group input:valid, .e-input-group.e-control-wrapper input:not(:valid), .e-input-
group.e-control-wrapper input:valid, .e-float-input.e-control-wrapper textarea:not(:valid), .e-float-
```

```
input.e-control-wrapper textarea:valid, .e-float-input textarea:not(:valid), .e-float-input textarea:valid,
.e-input-group.e-control-wrapper textarea:not(:valid), .e-input-group.e-control-wrapper textarea:valid,
.e-input-group textarea:not(:valid), .e-input-group textarea:valid {
```

```
font-size: 30px;
```

```
height: 40px;
```

```
}
```

```
,
```

### Customizing the TextBox placeholder

Use the following CSS to customize the TextBox placeholder

```
`css
```

```
/ To specify font size and color /
```

```
.e-float-input.e-control-wrapper:not(.e-error) input:valid ~ label.e-float-text, .e-float-input.e-control-
wrapper:not(.e-error) input ~ label.e-label-top.e-float-text {
```

```
color: pink;
```

```
font-size: 15px;
```

```
}
```

```
,
```

### How To

#### Set the rounded corner

Render the TextBox with **rounded corner** by adding the **e-corner** class to the input parent element.

### CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <div class="e-input-group e-corner">
 <input class="e-input" type="text" placeholder="Enter Date" />

 </div>
 <div class="e-float-input e-input-group e-corner">
 <input type='text' required />

 <label class="e-float-text">Enter Date </label>

 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
 focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
```



```

 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener("blur", function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
}
// Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
var inputIcon = document.querySelectorAll('.e-input-group-icon');
for (var i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[i].addEventListener('mouseup', function () {
 var element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
}
</script>
<style>
 .e-input-group.e-corner{
 border-radius: 4px;
 }
</style>

```

## ROUNDED-CORNER.CS

```

public ActionResult Index()
{
 return View();
}

```

## Set the disabled state

Disable the TextBox by adding the `e-disabled` to the input parent element and set `disabled` attribute to the input element.

## CSHTML

```

<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <input class="e-input" type="text" placeholder="Enter Name" disabled
 />
 <div class="e-float-input e-disabled">

```

```

 <input type='text' required disabled />

 <label class="e-float-text">Enter Name</label>
 </div>
</div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
 focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener('focus', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener('blur', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
 }
</script>

```

### DISABLED.CS

```

public ActionResult Index()
{
 return View();
}

```

### Set the read-only TextBox

You can make the TextBox as **read-only** by setting the **readonly** attribute to the input element.

### CSHTML

```

<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <input class="e-input" type="text" placeholder="Enter Name"
value="John" readonly />
 <div class="e-float-input">
 <input type='text' required readonly value="John" />

 <label class="e-float-text e-label-top">Enter Name</label>
 </div>
 </div>
</div>

```

```

</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input,.e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
 focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener('focus', function () {
 this.parentNode.classList.add('e-input-focus');
 });
 inputElement[i].addEventListener('blur', function () {
 this.parentNode.classList.remove('e-input-focus');
 });
 }
</script>

```

### READ-ONLY.CS

```

public ActionResult Index()
{
 return View();
}

```

### Add TextBox programmatically

- Create a TypeScript file and import the `Input` modules from `ej2-inputs` library as shown below.

`typescript

```
import {Input} from '@syncfusion/ej2-inputs';
```

,

- Pass the `HTML Input` element as parameter to the `createInput` method.
- You can also add the icons on the input by passing `buttons` property value with the class name `e-input-group-icon` as parameter to the `createInput` method.

### CSHTML

```

<div class="control_wrapper accordion-control-section">
 <div id="input-container">
 </div>
 </div>
 <script>
 ej.base.enableRipple(true);
 var element = document.createElement('input');
 document.getElementById('input-container').appendChild(element);
 new ej.inputs.Input.createInput({
 element: element,
 properties: {
 placeholder: 'Enter Name'
 }
 });
 </script>

```

```

 });
 var element2 = document.createElement('input');
 document.getElementById('input-container').appendChild(element2);
 new ej.inputs.Input.createInput({
 element: element2,
 buttons: ['e-input-group-icon e-input-down'],
 properties: {
 placeholder: 'Enter Value'
 }
 });
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener("focus", function () {
 this.parentNode.classList.add('e-input-focus')
 });
 inputElement[i].addEventListener("blur", function () {
 this.parentNode.classList.remove('e-input-focus')
 });
 }
 // Add 'e-input-btn-ripple' class to the icon element for achive ripple
effect when click on the icon.
 var inputIcon = document.querySelectorAll('.e-input-group-icon');
 for (var i = 0; i < inputIcon.length; i++) {
 inputIcon[i].addEventListener('mousedown', function () {
 this.classList.add('e-input-btn-ripple');
 });
 inputIcon[i].addEventListener('mouseup', function () {
 var element = this;
 setTimeout(function () {
 element.classList.remove('e-input-btn-ripple');
 }, 500);
 });
 }
</script>
<style>
 .e-input-group-icon:before {
 font-family: e-icons;
 }
 .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-
classes */
 content: "□";
 }
</style>

```

### TEXTBOX.CS

```

public ActionResult Index()
{
 return View();
}

```

### Add floating label to TextBox programmatically

The **Floating Label TextBox** floats label above the TextBox after focusing, or entering a value in the TextBox.

Floating label supports the types of actions as given below.

| Type   | Description                                                                                     |
|--------|-------------------------------------------------------------------------------------------------|
| Auto   | The floating label will float above the input after focusing, or entering a value in the input. |
| Always | The floating label will always float above the input.                                           |
| Never  | By default, never float the label in the input when the placeholder is available.               |

- Create a TypeScript file and import the **Input** modules from **ej2-inputs** library as shown below.

`typescript

```
import {Input} from '@syncfusion/ej2-inputs';
```

- Pass the **HTML Input** element and **floatLabelType** property as **Auto** to the **createInput** method.
- Set the **placeholder** value to the input element via **element** attribute or pass the parameter to the **createInput** method.

The **watermark label** will be updated based on the specified **placeholder** value of the **Floating Label TextBox**.

- You can add the **icons** on the input by passing **buttons** property value with the class name **e-input-group-icon** as parameter to the **createInput** method.

### CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <div id="input-container-1">
 <h4> FloatLabelType as Auto </h4>
 </div>
 <div id="input-container-2">
 <h4> FloatLabelType as Always </h4>
 </div>
 <div id="input-container-3">
 <h4> FloatLabelType as Never </h4>
 </div>
 <div id="input-container-4">
 <h4> Float label input with icons </h4>
 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 var inputObj;
```

```

var element = document.createElement('input');
document.getElementById('input-container-1').appendChild(element);
new ej.inputs.Input.createInput({
 element: element,
 floatLabelType: "Auto",
 properties: {
 placeholder: 'Enter Name'
 }
});
var element2 = document.createElement('input');
document.getElementById('input-container-2').appendChild(element2);
new ej.inputs.Input.createInput({
 element: element2,
 floatLabelType: "Always",
 properties: {
 placeholder: 'Enter Name'
 }
});
var element3 = document.createElement('input');
document.getElementById('input-container-3').appendChild(element3);
new ej.inputs.Input.createInput({
 element: element3,
 floatLabelType: "Never",
 properties: {
 placeholder: 'Enter Name'
 }
});
var element4 = document.createElement('input');
document.getElementById('input-container-4').appendChild(element4);
new ej.inputs.Input.createInput({
 element: element4,
 floatLabelType: "Auto",
 buttons: ['e-input-group-icon e-input-down'],
 properties: {
 placeholder: 'Enter Name'
 }
});
</script>
<style>
 .e-input-group-icon:before {
 font-family: e-icons;
 }
 .e-input-group-icon.e-input-down:before { /* csslint allow: adjoining-classes */
 content: "□";
 }
 #input-container-03 .e-input-group { /* csslint allow: adjoining-classes */
 margin: 30px 0;
 }
 #input-container-03 .e-float-input { /* csslint allow: adjoining-classes */
 margin: 30px 0;
 }
</style>

```

## FLOATING-LABEL.CS

```
public ActionResult Index()
{
 return View();
}
```

### Change the floating label color of the TextBox

You can change the floating label color of the TextBox for both **success** and **warning** validation states by applying the following CSS styles.

`CSS

*/ For Success state /*

```
.e-float-input.e-success label.e-float-text,
.e-float-input.e-success input:focus ~ label.e-float-text,
.e-float-input.e-success input:valid ~ label.e-float-text {
color: #22b24b;
}
```

*/ For Warning state /*

```
.e-float-input.e-warning label.e-float-text,
.e-float-input.e-warning input:focus ~ label.e-float-text,
.e-float-input.e-warning input:valid ~ label.e-float-text {
color: #ffca1c;
}
`
```

## CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <div class="e-float-input e-input-group e-success">
 <input type='text' required />

 <label class="e-float-text">Success</label>
 </div>
 <div class="e-float-input e-input-group e-warning">
 <input type='text' required />

 <label class="e-float-text">Warning</label>
 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
```

```
// Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener('focus', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener('blur', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
}
</script>
<style>
/* For Success state */
.e-float-input.e-success label.e-float-text,
.e-float-input.e-success input:focus ~ label.e-float-text,
.e-float-input.e-success input:valid ~ label.e-float-text {
 color: #22b24b;
}
/* For Warning state */
.e-float-input.e-warning label.e-float-text,
.e-float-input.e-warning input:focus ~ label.e-float-text,
.e-float-input.e-warning input:valid ~ label.e-float-text {
 color: #ffc107;
}
</style>
```

## VALIDATION-STATE.CS

```
public ActionResult Index()
{
 return View();
}
```

Output be like the below.

```
Success
Content
Warning
Content
```



Change the color of the TextBox based on its value

You can change the color of the TextBox by validating its value using regular expression in the `keyup` event for predicting the numeric values as demonstrated in the following code sample.

### CSHTML

```
<div class="control-section">
 <div class="wrap">
 <div class="e-input-group">
 <input class="e-input" id="numericOnly" type="text"
placeholder="Enter numeric values" onkeyup="onKeyUp(this)" />
 </div>
 <div class="e-float-input e-input-group">
 <input type="text" onkeyup="onKeyUp(this)" required />

 <label class="e-float-text">Enter numeric values</label>
 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener('focus', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.add('e-input-focus');
 } else {
 this.parentNode.classList.add('e-input-focus');
 }
 });
 inputElement[i].addEventListener('blur', function () {
 var parentElement = this.parentNode;
 if (parentElement.classList.contains('e-input-in-wrap')) {
 parentElement.parentNode.classList.remove('e-input-focus');
 } else {
 this.parentNode.classList.remove('e-input-focus');
 }
 });
 }
 function onKeyUp(e) {
 let str = e.value.match(/^[0-9]+$/);
 if (!((str && str.length > 0)) && e.value.length) {
 e.parentElement.classList.add('e-error');
 } else {
 e.parentElement.classList.remove('e-error');
 }
 }
</script>
<style>
 .e-input-group.e-error .e-input { /* csslint allow: adjoining-classes */
 color: #f44336;
 }
</style>
```

```
.e-float-input.e-error input { /* csslint allow: adjoining-classes */
 color: #f44336;
}
.wrap label { /* csslint allow: adjoining-classes */
 font-weight: bold;
}
</style>
```

## TEXT-COLOR.CS

```
public ActionResult Index()
{
 return View();
}
```

Output be like the below.

### Normal Input

Content

### Floating Input

Enter numeric values

## Add floating label to read-only textbox

You can achieve floating label for read-only textboxes by adding/removing `e-label-top` and `e-label-bottom` classes to the label element

In this sample, click the update value button to fill the read-only textbox with value and float a label.

## CSHTML

```
<div class="control-section">
 <div class="control_wrapper accordion-control-section">
 <div class="e-float-input">
 <input class="e-input myField" id="myText" name="readonlyAttr"
type="text" readOnly>

 <label class="e-float-text">Enter value</label>
 </div>
 @Html.EJS().Button("valuebtn").CssClass("e-btn
update_value").Content("Set value").Render()
 @Html.EJS().Button("removebtn").CssClass("e-btn
remove_value").Content("Remove value").Render()
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 document.getElementById('valuebtn').onclick = () => {
 (document.getElementsByClassName('myField')[0]).value = '10';
```

```

 checkFloatingLabel('myText')
 }
 document.getElementById('removebtn').onclick = () => {
 (document.getElementsByClassName('myField')[0]).value = '';
 checkFloatingLabel('myText')
 }
 function checkFloatingLabel(id) {
 var inputElement = document.getElementById(id);
 var labelElement = inputElement.parentElement.querySelector('.e-
float-text');
 if (inputElement.value !== '') {
 labelElement.classList.remove('e-label-bottom');
 labelElement.classList.add('e-label-top');
 } else {
 labelElement.classList.remove('e-label-top');
 labelElement.classList.add('e-label-bottom');
 }
 }
 var inputField = document.getElementById('myText');
</script>
<style>
 .wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 260px;
 }
 .update_value, .remove_value {
 margin-top: 20px;
 }
 .remove_value {
 margin-left: 10px;
 }
</style>

```

### FLOAT-READ-ONLY.CS

```

public ActionResult Index()
{
 return View();
}

```

Customize the textbox background-color and text-color

You can customize the textbox styles such as background-color, text-color and border-color by overriding its default styles.

**Note:** To change the styles of the **floating label**, you must override the style to the input element.

### CSHTML

```

<div class="control-section">
 <div class="wrap">
 <div class="e-input-group">
 <input class="e-input" type="text" placeholder="First Name" />
 </div>
 </div>
</div>

```

```

 <div class="e-float-input">
 <input type="text" required />

 <label class="e-float-text">Last Name</label>
 </div>
 </div>
</div>
<script>
 ej.base.enableRipple(true);
 // To get the all input fields and its container.
 var inputElement = document.querySelectorAll('.e-input-group .e-
input, .e-float-input.e-input-group input');
 // Add 'e-input-focus' class to the input for achive ripple effect when
 focus on the input field.
 for (var i = 0; i < inputElement.length; i++) {
 inputElement[i].addEventListener('focus', function () {
 this.parentNode.classList.add('e-input-focus');
 });
 inputElement[i].addEventListener('blur', function () {
 this.parentNode.classList.remove('e-input-focus');
 });
 }
</script>
<style>
 .wrap {
 box-sizing: border-box;
 margin: 0 auto;
 padding: 20px 10px;
 width: 260px;
 }

 .wrap label { /* csslint allow: adjoining-classes */
 font-weight: bold;
 }

 .wrap .e-float-input { /* csslint allow: adjoining-classes */
 margin: 30px 0;
 }

 .wrap .e-input-group { /* csslint allow: adjoining-classes */
 margin: 25px 0;
 }

 /* To change the background-color and text-color for textbox */
 .e-input-group,
 .e-float-input,
 .e-float-input.e-input-group { /* csslint allow: adjoining-classes */
 background: yellow;
 color: green;
 }

 /* To change the border-color of the textbox */
 .e-input-group:not(.e-success):not(.e-warning):not(.e-error):not(.e-
float-icon-left),
 .e-input-group:hover:not(.e-success):not(.e-warning):not(.e-
error):not(.e-float-icon-left) { /* csslint allow: adjoining-classes */
 border-color: #0800ff;
 }

 /* To change the border-color of the floating-label textbox */
 .e-float-input input,

```

```
.e-float-input:hover:not(.e-input-group):not(.e-success):not(.e-
warning):not(.e-error):not(.e-disabled) input:not([disabled]) { /* csslint
allow: adjoining-classes */
 border-color: #0800ff;
}
</style>
```

## TEXTBOX-CUSTOMIZE.CS

```
public ActionResult Index()
{
 return View();
}
```

## ASP.NET MVC textbox control

**Note:** From v16.3.21 version, the textbox is provided as ASP.NET MVC control to achieve the floating label textbox with minimal code. You can find the available textbox properties, methods, and events in the [API reference](#).

The following table describes the migration from CSS textbox to ASP.NET MVC textbox control.

### Normal textbox

| **Rendering mode** | **CSS textbox** | **ASP.NET MVC textbox control** |

|-----|-----|-----|

| Default rendering | <div class='e-input-group'><br/><input class='e-input' type='text'  
placeholder='Enter Value'/><br/></div> |

@Html.EJS().TextBox("default").<br/>Placeholder("Enter  
Value").<br/>FloatLabelType<br/>(FloatLabelType.Never).<br/>Render() |

| Textbox with clear button | <div class='e-input-group'><br/><input class='e-input'  
placeholder='Enter Value' required='true'><br/><span class='e-clear-  
icon'></span><br/></div><br/><br/> Note: You have to write action for clear button. |

@Html.EJS().TextBox("clear-input").Placeholder("Enter  
Value").ShowClearButton(true).<br/>FloatLabelType(FloatLabelType.<br/>Never).Render() |

| Validation states | <div class='e-input-group e-error'><br/><input class='e-input' type='text'  
placeholder='Enter Value' /><br/></div><br/><br/> Note: Textbox control consists of three types of  
validation rules such as success, warning, and error. | @Html.EJS().TextBox("clear-  
input").Placeholder("Enter Value").CssClass("e-  
error").FloatLabelType<br/>(FloatLabelType.Never).<br/>Render() |

### Floating label textbox

| **Rendering mode** | **CSS textbox** | **ASP.NET MVC textbox control** |

|-----|-----|-----|

| Default rendering | <div class='e-float-input'><br/><input type='text' required /><br/><span  
class='e-float-line'></span><br/><label class='e-float-text'>Enter Value</label><br/></div> |

@Html.EJS().TextBox("default").<br/>Placeholder("Enter  
Value").<br/>FloatLabelType<br/>(FloatLabelType.Auto).<br/>Render() |

| Textbox with clear button | `<div class='e-float-input e-input-group'><br/><input type='text' required value= 'Clear Input' /><br/><span class='e-float-line'></span><br/><label class='e-float-text'>Enter Value</label><br/><span class='e-clear-icon'></span><br/></div><br/><br/>`

Note: You have to write action for clear button. | `@Html.EJS().TextBox("clear-input").Placeholder("Enter Value").ShowClearButton(true).`

`<br/>FloatLabelType(FloatLabelType.Auto).Render() |`

| Validation states | `<div class='e-float-input e-error'><br/><input type='text' required /><br/><span class='e-float-line'></span><br/><label class='e-float-text'>Enter Value</label><br/></div><br/><br/>`

Note: Textbox control consists of three types of validation rules such as success, warning, and error. | `@Html.EJS().TextBox("clear-input").Placeholder("Enter Value").CssClass("e-error").FloatLabelType(FloatLabelType.Auto).Render() |`

## TimePicker

### Getting Started with ASP.NET MVC TimePicker Control

This section briefly explains about how to include [ASP.NET MVC TimePicker](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

`<namespaces>`

`<add namespace="Syncfusion.EJ2"/>`

</namespaces>

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

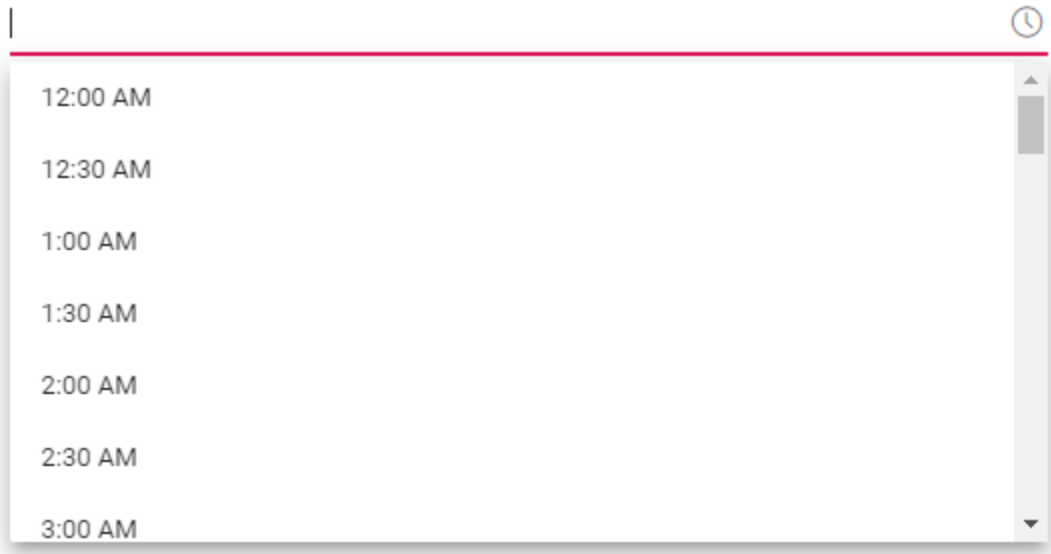
### Add ASP.NET MVC TimePicker control

Now, add the Syncfusion ASP.NET MVC TimePicker control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().TimePicker("timepicker").Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC TimePicker control will be rendered in the default web browser.



### Setting the value within min and max time

The following example demonstrates how to set the value, min and max time on initializing the TimePicker. The TimePicker allows you to select the time value within a range from 1:00 AM to 11:00 AM.

#### **CSHTML**

```
@Html.EJS().TimePicker("timepicker").Value(ViewBag.value).Min(ViewBag.minVal).Max(ViewBag.maxVal).Render()
```

#### **HOMECONTROLLER.CS**

```
public ActionResult Index()
{
 ViewBag.minVal = new DateTime(2022, 05, 07, 1, 00, 00);
 ViewBag.maxVal = new DateTime(2022, 05, 07, 11, 00, 00);
 ViewBag.value = new DateTime(2022, 05, 07, 4, 00, 00);
 return View();
}
```





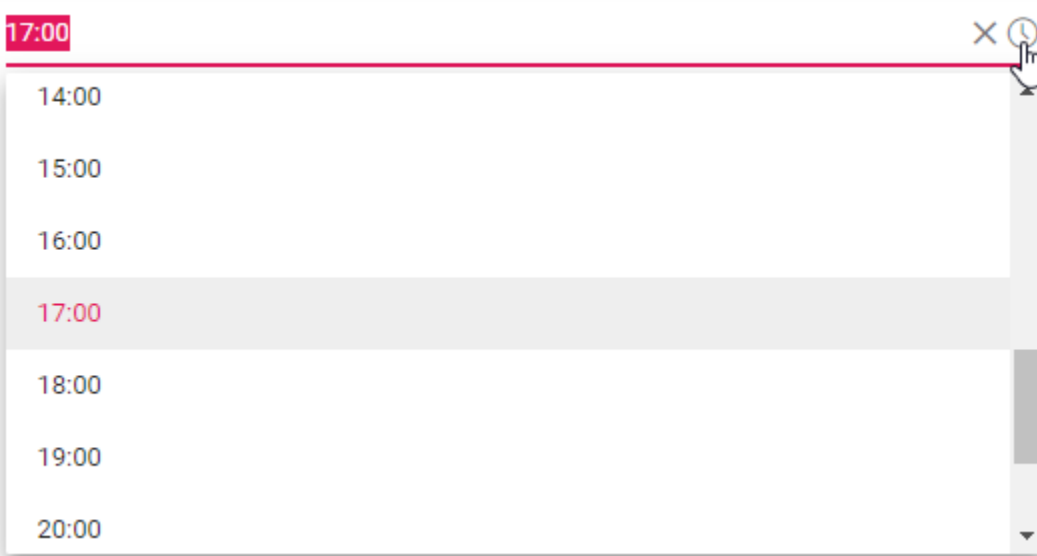
### Setting the time format

Time formats is a way of representing the time value in different string format in textbox and popup list. By default, the TimePicker's format is based on the culture. You can also customize the format by using the [Format](#) property. To know more about the time format standards, refer to the [Date and Time Format](#) section.

The following example demonstrates the TimePicker control in 24 hours format with 60 minutes interval. The time interval is set to 60 minutes by using the [Step](#) property.

### CSHTML

```
@Html.EJS().TimePicker("timepicker").Value(DateTime.Now).Format("HH:mm").Step(60).Render()
```



**Note:** [View Sample in GitHub.](#)

See also

- [Render TimePicker with min and max time](#)
- [How to achieve validation with TimePicker](#)
- [Render TimePicker with specific culture](#)
- [How to get and set value in TimePickerFor](#)

## Time Range

TimePicker provides an option to select a time value within a specified range by using the [min](#) and [max](#) properties. The min value should always be lesser than the max value.

When the min and max properties are configured and the selected time value is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

The value property depends on the min/max with respect to [strictMode](#) property.

The following example allows you to select a time value within a range of **9:00 AM** to **11:30 AM**.

### CSHTML

```
@Html.EJS().TimePicker("timepicker").Value(ViewBag.value).Min(ViewBag.minVal)
.Max(ViewBag.maxVal).Render()
```

### TIMERANGE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController: Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 ViewBag.minVal= new DateTime(2017,08,03, 09,00,00);
 ViewBag.maxVal = new DateTime(2017,08,03, 11,30,00);
 ViewBag.value = new DateTime(2017,08,03, 11,00,00);
 return View();
 }
 }
}
```

**Note:** If the value of **min** or **max** property is changed through code behind you have to update the **value** property to set within the range.

### Time Range customization using two TimePicker components

Here, two TimePicker components are used to select the start and end time. The below sample illustrates the appointment time selection scenario with the start and end time option.

Before the start time selection, the end time TimePicker is in disable state. When the start time is selected, then you will be able to select the end time or else, need to select the entire business hours 9:00 to 18:00 from the Business Hours option. Once the options are checked, both the TimePicker components goes to readonly state.

### CSHTML

```
<div class="control-section">
 <div class="control_wrapper">
 <div class="pane">
 <div class="tabs-wrap">
 <div class="wrap">

@Html.EJS().TimePicker("start").Created("onStartCreate").Placeholder("Start
Time").Change("onEnableEndTime").Render()

 </div>
 </div>
 <div class="tabs-wrap" style="clear: both">
 <div class="wrap">

@Html.EJS().TimePicker("end").Created("onEndCreate").Placeholder("End
Time").Enabled(false).Render()

 </div>
 </div>
 <div class="tabs-wrap" style="clear: both;padding: 14px
10px;">

 <div class="wrap">

@Html.EJS().CheckBox("dayRange").Change("changeTime").Label("Business
Hours").Render()

 </div>
 </div>
 </div>
 </div>
</div>
<script>
document.addEventListener('DOMContentLoaded', function () {
 isStartTimeChange = true;
 endInput = document.getElementById('end');
});
function onEnableEndTime(args) {
 /*Enables end time if start time is selected*/
 if (isStartTimeChange) {
 endTime.enabled = true;
 endTime.value = null;
 endInput.value = '';
 value = new Date(+args.value);
 value.setMinutes(value.getMinutes() + endTime.step);
 endTime.min = value;
 } else {
 isStartTimeChange = true;
 }
}
function changeTime() {
 /*To determine whether we have selected business hours or not*/
 let element = document.getElementById('dayRange');
 isStartTimeChange = false;
```

```

 if (element.checked) {
 /*Business hours*/
 startTime.value = new Date('9/6/2017 9:00');
 endTime.enabled = true;
 endTime.value = new Date('9/6/2017 18:00');
 startTime.readonly = true;
 endTime.readonly = true;
 } else {
 endTime.value = null;
 startTime.value = null;
 endInput.value = '';
 startTime.readonly = false;
 endTime.readonly = false;
 endTime.enabled = false;
 }
}
function onStartCreate() {
 startTime = document.getElementById('start').ej2_instances[0];
}
function onEndCreate() {
 endTime = document.getElementById('end').ej2_instances[0];
}
</script>

```

### TIMERANGE-CUSTOMIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController: Controller
 {
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
 }
}

```

### Enable the Masked Input

TimePicker has `enableMask` property that provides the option to enable the built-in date masking support.

### CSHTML

```
@Html.EJS().TimePicker("element").EnableMask("true").Render()
```

The mask pattern is defined based on the provided time format to the component. If the format is not specified, the mask pattern is formed based on the default format of the current culture.

| **Keys** | **Actions** |

| --- | --- |

| Up / Down arrows | To increment and decrement the selected portion of the time. |

| Left / Right arrows and Tab | To navigate the selection from one portion to next portion |

The following example demonstrates default and custom format of TimePicker component with mask.

#### **CSSHTML**

```
<div class="control-section">
 <div class="control_wrapper">
 <div class="pane">
 <div class="tabs-wrap">
 <div class="wrap">
 // Specifies the masked TimePicker without format
property.
@Html.EJS().TimePicker("element").EnableMask("true").Render()
 </div>
 </div>
 <div class="tabs-wrap">
 <div class="wrap">
 // Specifies the masked TimePicker with format.
@Html.EJS().TimePicker("element").EnableMask("true").Format("M/d/yyyy").Render()
 </div>
 </div>
 </div>
 </div>
</div>
```

#### **DATE-FORMAT.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController : Controller
 {
 {
 public ActionResult Sample()
 {
 return View();
 }
 }
 }
}
```

#### Configure Mask Placeholder

You can change mask placeholder value through property `maskPlaceholder`. By default , it takes the full name of time co-ordinates such as `hour`, `minute` and `second`.

While changing to a culture other than **English**, ensure that locale text for the concerned culture is loaded through load method of L10n class for mask placeholder values like below.

```
`typescript
//load the locale object to set the localized mask placeholder value
L10n.load({
'de': {
'timepicker': { hour: 'Stunde' ,minute: 'Minute', second:'Sekunde' }
}
});
`
```

The following example demonstrates default and customized mask placeholder value.

### CSHTML

```
<div class="control-section">
 <div class="control_wrapper">
 <div class="pane">
 <div class="tabs-wrap">
 <div class="wrap">
 // Specifies the masked TimePicker with format
property.
@Html.EJS().TimePicker("element").EnableMask("true").Render()
 </div>
 </div>
 <div class="tabs-wrap">
 <div class="wrap">
 // Specifies the masked TimePicker without format.
@Html.EJS().TimePicker("element").EnableMask("true").MaskPlaceholder(new
Syncfusion.EJ2.Calendars.TimePickerMaskPlaceholder {Hour="h", Minute="m",
Second="s"}).Render()
 </div>
 </div>
 </div>
 </div>
</div>
```

### MASK-PLACEHOLDER.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController : Controller
 {
 public ActionResult Sample()
 }
}
```

```

 {
 return View();
 }
}

```

## Globalization

Globalization is the combination of internalization and localization. You can adapt the component to various languages by parsing and formatting the date or number [internationalization](#) and also add culture specific customization and translation to the text [localization](#).

By default, TimePicker time format and meridian names are specific to the **American English** culture. It utilizes the [Essential JavaScript 2 Internationalization](#) package to parse and format the date object based on the culture by using the official [UNICODE CLDR](#) JSON data. It provides the `loadCldr` method to load culture specific CLDR JSON data.

- Set the culture by using the [locale](#) property.

To go with the different culture other than **English**, follow the below steps.

- Install the **CLDR-Data** package by using the below command (it installs the CLDR JSON data). To know more about CLDR-Data refer the [CLDR-Data](#) link.

```
npm install cldr-data --save
```

Once the package installed, you can find the culture specific JSON data under the location `/node_modules/cldr-data`.

In ASP.NET MVC refer the culture files directly from `/node_modules/cldr-data` location.

```

`typescript
function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numbers.json', 'timeZoneNames.json'];
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;
 ajax = new ej.base.Ajax(location.origin + location.pathname + '/../node_modules/cldr-data/main/' +
 name + '/' + files[prop], 'GET', false);
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 };
}

```

```

loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
loadCulture(prop);
}
}
`

```

- Before changing to a culture other than English, ensure that locale text for the concerned culture is loaded through `load` method of `L10n` class.

```

`typescript
var L10n = ej.base.L10n;
L10n.load({
'de': {
'timepicker': { placeholder: 'Wählen Sie Zeit' }
}
});
`

```

The following example demonstrates the TimePicker in German culture.

### CSHTML

```

@Html.EJS().TimePicker("timepicker").Placeholder("Select Time").Render()
<script>
 document.addEventListener('DOMContentLoaded', function () {
 timepicker = document.getElementById('timepicker').ej2_instances[0];
 var L10n = ej.base.L10n;
 L10n.load({
 'de': {
 'timepicker': { placeholder: 'Wählen Sie Zeit' }
 }
 });
 loadCultureFiles("de");
 timepicker.locale = 'de';
 });
 function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;
 if (name === 'ar' && prop === files.length - 1) {
 ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../..../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
 }
 };
 }

```



```

 } else {
 ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
};
for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
}
}
</script>

```

## GLOBALIZATION.CS

### Right-To-Left

The TimePicker supports RTL (right-to-left) functionality for languages like Arabic and Hebrew to displays the text in the right-to-left direction. Use [enableRtl](#) property to set the RTL direction.

The following code example demonstrates the TimePicker component in **Arabic** culture. It also explains how to set localized text to the placeholder using [L10n.load](#) method.

The following example demonstrates TimePicker in **Arabic** culture with right-to-left direction.

### CSHTML

```

@Html.EJS().TimePicker("timepicker").EnableRtl(true).Placeholder("Select a
time").Render()
<script>
 document.addEventListener('DOMContentLoaded', function () {
 timepicker = document.getElementById('timepicker').ej2_instances[0];
 var L10n = ej.base.L10n;
 L10n.load({
 'ar': {
 'timepicker': { placeholder: 'حدد الوقت' }
 }
 });
 loadCultureFiles("ar");
 timepicker.locale = "ar";
 });
 function loadCultureFiles(name) {
 var files = ['ca-gregorian.json', 'numbers.json',
'timeZoneNames.json'];
 if (name === 'ar') {
 files.push('numberingSystems.json');
 }
 var loader = ej.base.loadCldr;
 var loadCulture = function (prop) {
 var val, ajax;

```

```

 if (name === 'ar' && prop === files.length - 1) {
 ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../../node_modules/cldr-data/supplemental/' + files[prop], 'GET',
false);
 } else {
 ajax = new ej.base.Ajax(location.origin + location.pathname
+ '/../../node_modules/cldr-data/main/' + name + '/' + files[prop], 'GET',
false);
 }
 ajax.onSuccess = function (value) {
 val = value;
 };
 ajax.send();
 loader(JSON.parse(val));
 };
 for (var prop = 0; prop < files.length; prop++) {
 loadCulture(prop);
 }
}
</script>

```

### RTL.CS

### Strict mode in TimePicker Control

The [strictMode](#) is an act that allows you to enter only valid time value within the specified min/max range in the textbox. If the time value is invalid, the control value sets to the previous value. If the time value is out of range, the control sets the time value to min/max value.

The following example demonstrates the TimePicker in `strictMode` with min/max range of 10:00 AM to 4:00 PM . It allows you to enter only valid time within the specified range. If you enter the out-of-range value like 8:00 PM, the value sets to the max time 4:00 PM as the value 8:00 PM is greater than max value of 4:00 PM. If you enter invalid time value like 9:00 tt, the value sets to the previous value.

### CSHTML

```

@Html.EJS().TimePicker("timepicker").Value(ViewBag.value).Min(ViewBag.minVal)
.Max(ViewBag.maxVal).StrictMode(true).Render()

```

### STRICTMODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController: Controller
 {
 public ActionResult DefaultFunctionalities()
 {

```

```

 ViewBag.minVal= new DateTime(2017,08,03, 10,00,00);
 ViewBag.maxVal = new DateTime(2017,08,03, 16,00,00);
 ViewBag.value = new DateTime(2017,08,03, 03,00,00);
 return View();
 }
}

```

By default, the TimePicker act in strictMode **false** state, that allows to enter the invalid or out-of-range time in textbox.

If the time is out-of-range or invalid, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

The following example demonstrates the **strictMode** as **false**. Here, it allows to enter the valid or invalid value in textbox. If you are entering the out-of-range or invalid time value, then the model value will be set to **out of range** time value or **null** respectively with highlighted **error** class to indicates the time is out of range or invalid.

### CSHTML

```

@Html.EJS().TimePicker("timepicker").Value(ViewBag.value).Min(ViewBag.minVal)
.Max(ViewBag.maxVal).StrictMode(false).Render()

```

### NORMALMODE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController: Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 ViewBag.minVal= new DateTime(2017,08,03, 10,00,00);
 ViewBag.maxVal = new DateTime(2017,08,03, 16,00,00);
 ViewBag.value = new DateTime(2017,08,03, 3,00,00);
 return View();
 }
 }
}

```

**Note:** If the value of **min** or **max** property is changed through code behind, update the **value** property to set within the range.

### Accessibility

The web accessibility makes web applications and its content more accessible to people with disabilities without any barriers. It especially it tracks the dynamic value changes and DOM changes.

The TimePicker component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the TimePicker component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2 Support](#) |  |

| [Section 508 Support](#) |  |

| [Screen Reader Support](#) |  |

| [Right-To-Left Support](#) |  |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### WAI-ARIA attributes

The TimePicker control has covered the [WAI-ARIA](#) specifications with the following list of WAI-ARIA attributes: `aria-haspopup`, `aria-selected`, `aria-disabled`, `aria-activedescendant`, `aria-expanded`, `aria-owns`, and `aria-autocomplete`.

Here in TimePicker, the `combobox` plays the role of input element, and the `listbox` plays the role of popup element.

- **Aria-haspopup:** Provides the information about whether this element display a pop-up window or not.
- **Aria-selected:** Indicates the current selected value of the TimePicker control.
- **Aria-disabled:** Indicates disabled state of the TimePicker control.
- **Aria-expanded:** Indicates the expanded state of the popup.
- **Aria-autocomplete:** Indicates whether user input completion suggestions are provided or not.
- **Aria-owns:** Attribute that creates a parent/child relationship between two DOM element in the accessibility layer.
- **Aria-activedescendent:** Attribute that helps in managing the current active child of the TimePicker control.
- **Role:** Attribute that gives assistive technology information for handling each element in a widget.

### Keyboard Interaction

Keyboard accessibility is one of the most important aspects of web accessibility. Disabled people like blind and those who have motor disabilities or birth defects use keyboard shortcuts more than the mouse.

The TimePicker control has built-in keyboard accessibility support by following the [WAI-ARIA practices](#).

**Note:** It supports the following list of shortcut keys to interact with the TimePicker control.

| Press | To do this |

| --- | --- |

| Upper Arrow | Navigate and select the previous item. |

| Down Arrow | Navigate and select the next item. |

| Left Arrow | Move the cursor towards arrow key pressed direction. |

| Right Arrow | Move the cursor towards arrow key pressed direction. |

| Home | Navigate and select the first item. |

| End | Navigate and select the last item. |

| Enter | Select the currently focused item and close the popup. |

| Alt + Upper Arrow | Close the popup. |

| Alt + Down Arrow | Open the popup. |

| Esc | Close the popup. |

In the below sample use the `alt+t` keys to focus the TimePicker control.

## CSHTML

```
<script>
 document.onkeyup = function (e) {
 var timepicker =
document.getElementById('timepicker').ej2_instances[0];
 if (e.altKey && e.keyCode === 84 /* t */) {
 // press alt+t to focus the control by calling public method.
 timepicker.focusIn(e);
 }
 };
</script>
@Html.EJS().TimePicker("timepicker").Placeholder("Select a time").Render()
```

## ACCESSIBILITY.CS

### Ensuring accessibility

The TimePicker component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TimePicker component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TimePicker component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Style and appearance in TimePicker Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the appearance of TimePicker wrapper element

Use the following CSS to customize the appearance of wrapper element.

`css

*/ To specify height and font size /*

```
.e-input-group input.e-input, .e-input-group.e-control-wrapper input.e-input, .e-input-group textarea.e-
input, .e-input-group.e-control-wrapper textarea.e-input {
```

```
font-size: 20px;
```

```
height: 40px;
```

```
}
```

,

#### Customizing the TimePicker icon element

Use the following CSS to customize the TimePicker icon element

```
`css
/ To specify background color and font size /
.e-time-wrapper .e-time-icon.e-icons, *.e-control-wrapper.e-time-wrapper .e-time-icon.e-icons {
font-size: 20px;
background-color: beige;
}
`
```

### Customizing the TimePicker popup

Use the following CSS to customize the TimePicker popup

```
`css
/ To specify height /
.e-timepicker.e-popup {
height: 100px;
}
`
```

### Customizing the TimePicker popup content

Use the following CSS to customize the TimePicker popup content

```
`css
/ To specify height /
.e-timepicker.e-popup .e-list-parent.e-ul li.e-list-item {
background-color: beige;
font-size: 20px;
}
`
```

### Full screen mode support in mobiles and tablets

The TimePicker full-screen mode feature enables users to view the popup element in full-screen mode on mobile devices with improved visibility and a better user experience. It is important to mention that this feature is exclusively available for mobile and tablet devices in both landscape and portrait orientations. To activate the full screen mode within the TimePicker, simply set the [FullScreenMode](#) API value to `true`. This action will extend the popup element to occupy the entire screen on mobile devices.

**Default Sample**

12:00 AM







## How To

### CSS customization

TimePicker allows you to customize the textbox and popup list appearance to suit your application by using [cssClass](#) property.

The below sample demonstrates customization of text appearance in a textbox, popup button, and popup list along with hover and active state by using `e-custom-style` class. Following is the list of available classes used to customize the entire TimePicker component.

| Class Name                                  | Description                                            |
|---------------------------------------------|--------------------------------------------------------|
| ---                                         | ---                                                    |
| <code>e-time-wrapper</code>                 | Applied to TimePicker wrapper element.                 |
| <code>e-timepicker</code>                   | Applied to input element and TimePicker popup element. |
| <code>e-time-wrapper.e-timepicker</code>    | Applied to input element only.                         |
| <code>e-input-group-icon.e-time-icon</code> | Applied to popup button.                               |
| <code>e-float-text</code>                   | Applied to floating label text element.                |
| <code>e-popup</code>                        | Applied to popup element.                              |
| <code>e-timepicker.e-popup</code>           | Applied to TimePicker popup element only.              |
| <code>e-list-parent</code>                  | Applied to popup UL element.                           |
| <code>e-timepicker.e-list-parent</code>     | Applied to TimePicker popup UL element only.           |
| <code>e-list-item</code>                    | Applied to LI elements.                                |
| <code>e-hover</code>                        | Applied to LI element hovering time.                   |
| <code>e-active</code>                       | Applied to active LI element.                          |

### CSHTML

```
@Html.EJS().TimePicker("timepicker").CssClass("e-custom-style").Placeholder("Select a time").Render()
<style>
 /*customize the input element text color*/
 .e-time-wrapper.e-custom-style #element { /* csslint allow: adjoining-classes */
 display: block;
 color: blue;
 }
 /*customize the floating label and popup button text color*/
 .e-time-wrapper.e-custom-style .e-float-text.e-label-bottom, /* csslint allow: adjoining-classes */
 .e-time-wrapper.e-custom-style .e-input-group-icon.e-time-icon.e-icons {
 /* csslint allow: adjoining-classes */
 color: blue;
 }
 /*customize the input element text selection*/
 .e-time-wrapper.e-custom-style.e-input-group::before, /* csslint allow: adjoining-classes */
 .e-time-wrapper.e-custom-style.e-input-group::after, /* csslint allow: adjoining-classes */
```

```

.e-time-wrapper.e-custom-style.e-input-group .e-timepicker::selection {
/* csslint allow: adjoining-classes */
 background: blue;
}
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul, /* csslint
allow: adjoining-classes */
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-item {
/* csslint allow: adjoining-classes */
 background-color: #c0ebff;
}
/*customize the list item hover color*/
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-
item.e-hover, /* csslint allow: adjoining-classes */
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-
item.e-active.e-hover { /* csslint allow: adjoining-classes */
 background-color: blue;
 color: #fff;
}
/*customize the active element text color*/
.e-timepicker.e-popup.e-custom-style .e-list-parent.e-ul .e-list-
item.e-active { /* csslint allow: adjoining-classes */
 color: #333;
 background-color: #fff;
}
}
</style>

```

## CSS.CS

### Client side validation using FormValidator

To achieve client side validation in a TimePicker component, use [Essential JavaScript 2 FormValidator](#). It provides an option to customize feedback error messages to the corresponding fields for taking action and resolving the issue.

In the following example, the required field validation is implemented by mapping the name attribute value to the rules property. It validates the TimePicker component and displays the validation message when the textbox value is empty during form post back or focus out.

## CSHTML

```

<form id="form-element" class="form-vertical">
 <div class="form-group">
 <div class="col-sm-3 control-label">Required</div>
 <div class="col-sm-6">
 @Html.EJS().TimePicker("formTest").Value(ViewBag.value).Render()
 </div>
 </div>
</form>
<script>
 document.addEventListener('DOMContentLoaded', function () {
 var options = {
 rules: {
 //must specify the name attribute value in rules section
 }
 }
 });

```

```

 'formTest': { required: true }
 },
 customPlacement: (inputElement, errorElement) => {
 //to place the error message in custom position
 //inputElement - target element where the error text will be
 appended
 //errorElement - error text which will be displayed.
 inputElement.parentElement.parentElement.appendChild(errorElement);
 }
 };
 var formObject = new ej.inputs.FormValidator('#form-element',
options);
 });
</script>

```

### VALIDATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class HomeController: Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 ViewBag.value = DateTime.Now;
 return View();
 }
 }
}

```

### Render TimePickerFor

The TimePickerFor component can be rendered by passing a value from the model. The selected date value can be retrieved during form submission using the post method at the server end.

The following sample demonstrates how to retrieve the value in the controller by rendering the TimePickerFor component.

### CSHTML

```

@model EJ2CoreSampleBrowser.Controllers.TimePicker
@using (Html.BeginForm())
{
 @Html.EJS().TimePickerFor(model => model.value).Render()
 <div>
 @Html.ValidationMessageFor(model => model.value)
 </div>
 <div id="submitButton">
 @Html.EJS().Button("btn").Content("Post").Render()
 </div>
}

```


```
}
```

**TIMEPICKERFOR.CS**

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
namespace EJ2CoreSampleBrowser.Controllers
{
 public class TimePicker
 {
 [Required(ErrorMessage = "Please enter the value")]
 public DateTime? value { get; set; }
 }
 public class HomeController : Controller
 {
 TimePicker TimePickerValue = new TimePicker();
 public ActionResult Index()
 {
 TimePickerValue.value = new DateTime(2020, 03, 03, 10, 00, 00);
 return View(TimePickerValue);
 }
 [HttpPost]
 public ActionResult Index(TimePicker model)
 {
 //posted value is obtained from the model
 TimePickerValue.value = model.value;
 return View(TimePickerValue);
 }
 }
}
```

Selected value will be retrieved as below.

```
public class HomeController : Controller
{
 TimePicker TimePickerValue = new TimePicker();
 0 references
 public ActionResult Index()
 {
 TimePickerValue.value = new DateTime(2020, 03, 03, 10, 00, 00);
 return View(TimePickerValue);
 }
 [HttpPost]
 0 references
 public ActionResult Index(TimePicker model)
 {
 //posted value is obtained from the model
 ▶ TimePickerValue.value = model.value;
 return View(TimePickerValue);
 }
}
```

▶  model.value | {10/22/2021 10:00:00 AM} ➡

## Toast

### Getting Started with ASP.NET MVC Toast Control

This section briefly explains about how to include [ASP.NET MVC Toast](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### **~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{ site.ej2version
 }/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

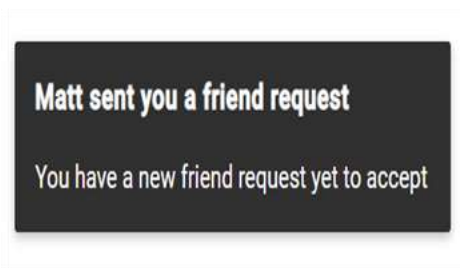
Add ASP.NET MVC Toast control

Now, add the Syncfusion ASP.NET MVC Toast control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to accept").Render()
<script type="text/javascript">
 setTimeout(() => {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
</script>
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Toast control will be rendered in the default web browser.



**Note:** [View Sample in GitHub](#).

See also

- [Real time example using Toast](#)
- [How to close the toast with click/tap](#)
- [How to prevent duplicate toast display](#)
- [How to show different types of toast](#)

## Configuring Options

This section explains the steps required to customize the appearance of the toast using built-in APIs.

### Title and content template

Toast can be created with the notification message. The message contains [Title](#) and [Content](#) of the toasts. The title and contents are adaptable in any resolution.

**Note:** The Title or Content property can be given as HTML element/element ID to a string that can be displayed as a toast.

### CSHTML

```
<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to accept").Render()
 @Html.EJS().Button("button").Content("Show Toast").CssClass("e-
btn").Render()
</div>
<script type="text/javascript">
 setTimeout(() => {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("button").addEventListener('click', function ()
 {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
</script>
```

### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### Specifying custom target

By default, the toast can be rendered in the document body. You can change the target position for toast rendering using the [Target](#) property. Based on the target, the [Position](#) will be updated.

### Close button

By default, the [ShowCloseButton](#) is not enabled. You can enable it by setting the true value. Before expiring the toast, you can use this button to close or destroy toasts manually.

### Progress bar

By default, the [ShowProgressBar](#) is not enabled. If it is enabled, it can visually indicate how long to get toast expires. Based on the [TimeOut](#) property, progress bar will appear.

### Progress bar direction

By default, the [ProgressDirection](#) is set to "Rtl" and it will appear from right to left direction. You can change the progressDirection to "Ltr" to make it appear from left to right direction.

### Newest on top

By default, the newly created toasts will append next with existing toasts. You can change the sequence like inserting before the toast by enabling the [NewestOnTop](#).

Here, the following sample demonstrates the combination of the [Target](#), [ShowCloseButton](#), [ShowProgressBar](#), and [NewestOnTop](#) properties in toast.

### CSHTML

```
<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("File Downloading").Position(p =>
p.X("Center")).ShowCloseButton(true).Target("#toast_target").NewestOnTop(true)
.ShowProgressBar(true).ProgressDirection("Ltr").ContentTemplate(@<div
class='progress'>

</div>).Render()
 @Html.EJS().Button("button").Content("Show Toast").CssClass("e-
btn").Render()
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("button").addEventListener('click', function ()
 {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
</script>
<style>
.e-toast-message {
 width: 100%;
}
.progress {
 height: 20px;
 position: relative;
 margin: 20px 0 20px 0;
 background: #555;
 -moz-border-radius: 25px;
 -webkit-border-radius: 25px;
```



```

 border-radius: 25px;
 box-shadow: inset 0 -1px 1px rgba(255, 255, 255, 0.3);
 }
 .progress span {
 background-color: #f0a3a3;
 background-image: -webkit-gradient(linear, left top, left bottom,
color-stop(0, #f0a3a3), color-stop(1, #f42323));
 display: block;
 height: 100%;
 border-radius: 10px;
 width: 50%;
 position: relative;
 overflow: hidden;
 }
 .progress span::after {
 background-image: -webkit-gradient(linear, 0 0, 100% 100%, color-
stop(.25, rgba(255, 255, 255, .2)), color-stop(.25, transparent), color-
stop(.5, transparent), color-stop(.5, rgba(255, 255, 255, .2)), color-
stop(.75, rgba(255, 255, 255, .2)), color-stop(.75, transparent),
to(transparent));
 content: "";
 position: absolute;
 top: 0;
 left: 0;
 bottom: 0;
 right: 0;
 background-size: 50px 50px;
 -webkit-animation: moveAnimate 2s linear infinite;
 overflow: hidden;
 }
 @@-webkit-keyframes moveAnimate {
 0% {
 background-position: 0 0;
 }
 100% {
 background-position: 50px 50px;
 }
 }
</style>

```

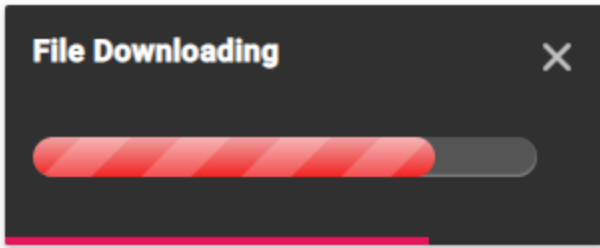
### **CONTROLLER.CS**

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.



### Width and height

The dimensions of the toast can be set using the [Width](#) and [Height](#) properties. This will individually set all toasts. You can create different custom dimension toasts.

By default, the toast can be rendered with '300px' width with 'auto' height.

**Note:** In mobile devices, the default width of the toast gets '100%' width of the page.

When you set toast width as '100%', the toast occupies full width and will be displayed at the top or bottom based on the position **Y** property.

Both the width and height properties allow setting pixels/numbers/percentage. The number value is considered as pixels.

### CSHTML

```
<div style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to
accept").Width("400").Height("150").Position(p =>
p.X("Center").Y("Bottom")).Render()
 <div class='row' style="padding-top: 20px" id="toast_pos_target">
 <table>
 <tr>
 <td>
 <div style='padding:25px 0 0 0;'>
 @Html.EJS().RadioButton("topAlign").Label("Top").Name("toast").Value("Target")
 .Change("checkboxChange").Render()
 </div>
 </td>
 <td>
 <div style='padding:25px 0 0 0;'>
 @Html.EJS().RadioButton("bottomAlign").Label("Bottom").Name("toast").Value("
Global").Change("checkboxChange1").Checked(true).Render()
 </div>
 </td>
 </tr>
 <tr>
 <td>
 <div style='padding:25px 0 0 0;'>
 @Html.EJS().CheckBox("fullWidth").Change("onChange").Label("100%
Width").Render()
 </div>
 </td>
 </tr>
 </table>
 </div>
```

```

 </td>
 </tr>
</table>
</div>
@Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
 var timeDelay = 1000;
 function checkboxChange(e) {
 var toastObj = document.getElementById('element').ej2_instances[0];
 if (e.event.target.checked) {
 toastObj.position.Y = "Top";
 toastObj.hide();
 toastShow(timeDelay);
 }
 }
 function checkboxChange1(e) {
 var toastObj = document.getElementById('element').ej2_instances[0];
 if (e.event.target.checked) {
 toastObj.position.Y = "Bottom";
 toastObj.hide();
 toastShow(timeDelay);
 }
 }
 function toastShow(timeOutDelay) {
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.show();
 }, timeOutDelay);
 }
 function onChange(e) {
 var toastObj = document.getElementById('element').ej2_instances[0];
 if (e.checked) {
 toastObj.hide();
 toastObj.width = "100%";
 toastObj.title = "";
 toastObj.content = "<div class='e-custom'>Take a look at our
next generation Javascript LEARN
MORE</div>";
 toastShow(timeDelay);
 } else {

```

```

 toastObj.hide();
 toastObj.width = 300;
 toastObj.title = 'Matt sent you a friend request',
 toastObj.content = 'You have a new friend request yet to
accept',
 toastShow(timeDelay);
 }
}
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## See Also

- [Prevent duplicate toasts](#)
- [Customize the progress bar](#)

## Positions

The toast position can be updated based on predefined positions or customizable positions. The predefined position combinations are updated in the **X** and **Y** [Position](#) properties.

### Predefined

#### X Positions

- Left
- Center
- Right

#### Y Positions

- Top
- Bottom

**Note:** In multiple toast display, the new toast position will not be updated on dynamic change of property values until the old toast messages removed.

<br/> The toast occupies full width when you set the width to '100%', so the X positions will not affect the changes when the width is '100%'.

### Custom

Custom **X** and **Y** positions can be given as pixels/numbers/percentage. The number value is considered as pixels based on the top and left values updated in the toast.

CSHTML

```

<div class="col-lg-12 control-section toast-pos-section">
 <div class="control_wrapper" id="toast_pos_target">
 @Html.EJS().Toast("toast_pos").Title("Matt sent you a friend
request").Content("You have a new friend request yet to accept").Icon("e-
laura").Position(p => p.X("Right").Y("Bottom")).Render()
 <div id="toast_pos_property">
 <table style="width: 100%;">
 <tbody>
 <tr>
 <td>
 <div style="padding:25px 0 0 0;">
 @Html.EJS().RadioButton("dropdownRadio").Label("Position").Name("toastPos").
Value("Position").Change("checkboxChange2").Checked(true).Render()
 </div>
 </td>
 <td>
 <div style="padding:25px 0 0 0;">
 @Html.EJS().RadioButton("customRadio").Label("Custom").Name("toastPos").Valu
e("Custom").Change("checkboxChange3").Render()
 </div>
 </td>
 </tr>
 <tr>
 <td colspan="2">
 <div id="dropdownChoose" style="padding-top:25px;">
 @Html.EJS().DropDownList("position").Fields(e=>e.Value("Id").Text("Value")).
Placeholder("Select a
position").PopupHeight("200px").Width("300px").Index(5).DataSource((IEnumerable<object>)
ViewBag.data).Change("valueChange").Render()
 </div>
 </td>
 </tr>
 <tr>
 <td colspan="2" id="customChoose" style="display:
none">
 <form id="formId" class="form-horizontal">
 <div class="e-row">
 <div class="e-float-input">
 <input class="e-input" id="xPos"
name="Digits" value="50" digits="true" data-digits-message="Please enter
digits only." required="">

 <label class="e-float-text"
for="Digits">X Position</label>
 </div>
 </div>
 <div class="e-row">
 <div class="e-float-input">
 <input class="e-input" id="yPos"
name="Digits" value="50" digits="true" data-digits-message="Please enter
digits only." required="">


```

```

<label class="e-float-text"
for="Digits">Y Position</label>
</div>
</div>
</form>
</td>
</tr>
<tr>
<td>
<div style="padding:25px 0 0 0;"

@Html.EJS().RadioButton("radio1").Label("Target").Name("toast").Value("Targe
t").Change("checkboxChange").Render()
</div>
</td>
<td>
<div style="padding:25px 0 0 0;"

@Html.EJS().RadioButton("radio2").Label("Global").Name("toast").Value("Globa
l").Change("checkboxChange1").Checked(true).Render()
</div>
</td>
</tr>
</tbody>
</table>
<div id="toast_btn" style="padding-top: 25px">
<button class="e-btn e-control" id="show_Toast"
style="margin-right: 15px"> Show Toast </button>
<button class="e-btn e-control" id="hideTosat"> Hide All
</button>
</div>
</div>
</div>
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 200);
 function toastShow() {
 setTimeout(
 () => {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 toastObj.show();
 }, 200);
 }
 var btnEle = document.getElementById('show_Toast');
 btnEle.onclick = function () {
 if (customFlag) {
 setcustomPosValue();
 }
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];

```

```

 toastObj.show();
 };
 document.getElementById('hideTosat').onclick = function () {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 toastObj.hide('All');
 };
 document.onclick = function (e) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 if (e.target !== btnEle && toastObj.target === document.body) {
 toastObj.hide('All');
 }
 };
 var customFlag = false;
 function checkboxChange(e) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 if (this.checked) {
 toastObj.hide('All');
 toastObj.target = document.getElementById('toast_pos_target');
 toastShow(1000);
 }
 }
 function checkboxChange1(e) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 if (this.checked) {
 toastObj.hide('All');
 toastObj.target = document.body;
 toastShow(1000);
 }
 }
 function checkboxChange2(e) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 var listObj = document.getElementById('position').ej2_instances[0];
 if (this.checked) {
 toastObj.hide('All');
 document.getElementById('dropdownChoose').style.display =
'table-cell';
 document.getElementById('customChoose').style.display = 'none';
 setToastPosValue(listObj.value.toString()); customFlag = false;
 toastShow(1000);
 }
 }
 function checkboxChange3(e) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 if (this.checked) {
 toastObj.hide('All');
 document.getElementById('dropdownChoose').style.display =
'none';
 document.getElementById('customChoose').style.display = 'table-
cell';
 setcustomPosValue(); customFlag = true; toastShow(1000);
 }
 }

```

```

 }
 //Setting Toast Custom Position
 function setcustomPosValue() {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 var initialWid = toastObj.width.toString();
 toastObj.width = initialWid;
 toastObj.position.X =
parseInt((document.getElementById('xPos')).value, 10);
 toastObj.position.Y =
parseInt((document.getElementById('yPos')).value, 10);
 }
 //Setting Toast Position
 function setToastPosValue(value) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 var initialWid = toastObj.width.toString();
 toastObj.width = initialWid;
 switch (value) {
 case 'topleft':
 toastObj.position.X = 'Left'; toastObj.position.Y = 'Top';
break;
 case 'topright':
 toastObj.position.X = 'Right'; toastObj.position.Y = 'Top';
break;
 case 'topcenter':
 toastObj.position.X = 'Center'; toastObj.position.Y = 'Top';
break;
 case 'topfullwidth':
 toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Top'; break;
 case 'bottomleft':
 toastObj.position.X = 'Left'; toastObj.position.Y =
'Bottom'; break;
 case 'bottomright':
 toastObj.position.X = 'Right'; toastObj.position.Y =
'Bottom'; break;
 case 'bottomcenter':
 toastObj.position.X = 'Center'; toastObj.position.Y =
'Bottom'; break;
 case 'bottomfullwidth':
 toastObj.width = '100%'; toastObj.position.X = 'Center';
toastObj.position.Y = 'Bottom'; break;
 }
 }
 function valueChange(e) {
 var toastObj =
document.getElementById('toast_pos').ej2_instances[0];
 toastObj.hide('All'); setToastPosValue(e.value.toString());
toastShow(1000);
 }
</script>
<style>
 .toast-pos-section #toast_pos_property {
 height: 500px;
 border: none;
 margin: auto;
 }

```



```

 }
 #toast_pos_property td {
 width: 50%;
 }
 .e-toast-icon.e-laura.e-icons {
 border-radius: 50%;
 background-image:
url('https://ej2.syncfusion.com/demos/src/toast/resource/laura.png');
 background-repeat: no-repeat;
 background-size: cover;
 height: 50px !important;
 width: 100px !important;
 background-size: 50px 50px;
 margin: 0;
 }
 @@media (min-width: 740px) {
 #toast_pos_property {
 width: 450px;
 }
 }
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.data = new Position().Positions();
 return View();
 }
 public class Position
 {
 public string Value { get; set; }
 public string Id { get; set; }
 public List<Position> Positions()
 {
 List<Position> position = new List<Position>
 {
 new Position { Id = "topleft", Value = "Top Left" },
 new Position { Id = "topright", Value = "Top Right" },
 new Position { Id = "topcenter", Value = "Top Center" },
 new Position { Id = "topfullwidth", Value = "Top Full
Width" },
 new Position { Id = "bottomleft", Value = "Bottom Left"
},
 new Position { Id = "bottomright", Value = "Bottom
Right" },
 new Position { Id = "bottomcenter", Value = "Bottom
Center" },
 new Position { Id = "bottomfullwidth", Value = "Bottom
Full Width" }
 };
 return position;
 }
 }
}

```

```
}

```

See Also

- [How to add dynamic template](#)

## Action Buttons

You can include action buttons to the toast control by adding the [Buttons](#) property. The collection of Syncfusion ASP.NET MVC button models can be bound to the `model` property inside the `Buttons` property. You can also include the click event callback function for each button.

### CSHTML

```
<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Position(p =>
 p.X("Right").Y("Bottom")).Buttons(item=> {
 item.Click("btnClick").ModelValue(ViewBag.ToastButtons1).Add();
 item.ModelValue(ViewBag.ToastButtons2).Add();
 }).Width("300").Height("150").ContentTemplate(@<div>
 <p>

 Anjolie Stokes
 </p>
 <div class="content">
 <p>Thanks for update!</p>
 </div>
 </div>).Render()
 <div id='templateToast' style="display: none;color:red"> System affected
by virus !!! </div>
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
 function btnClick(e) {
 var toastEle = ej.base.closest(e.target, '.e-toast');
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.hide(toastEle);
 }
</script>
<style>
```

```

.toast-img {
 width: 40px;
 height: 40px;
}
.name {
 padding-left: 20px;
 font-size: 17px;
 font-weight: 500;
}
.content {
 padding-left: 60px;
 font-size: 12px;
}
</style>

```

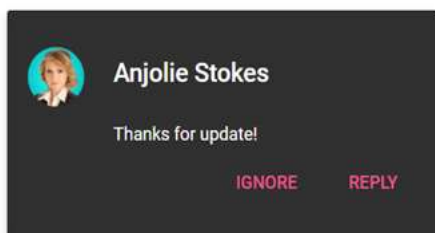
### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.ToastButtons1 = new DefaultButtonModel() { content =
"Ignore" };
 ViewBag.ToastButtons2 = new DefaultButtonModel() { content = "reply"
};
 return View();
 }
 public class DefaultButtonModel
 {
 public string content { get; set; }
 }
}

```

Output be like the below.



See Also

- [How to add dynamic template](#)

### Time out

The toast can be expired based on the [TimeOut](#) property. The toast can live till the time out reaches without user interaction, a time out value is considered as a millisecond.

- The `TimeOut` delay can be visually represented using `Progress Bar`.

- The [ExtendedTimeOut](#) property determines how long the toast should be displayed after a user hovers over it.

**Note:** You can terminate the process by using the [ShowCloseButton](#) property for destroying the toast at any time.

### CSHTML

```
<div style="width:400px; margin:0 auto;">
 <div class="e-float-input"><input class="e-input" id="toast_input_index"
required="" value="0"><label class="e-
float-text">Enter timeOut</label></div>
 @Html.EJS().Toast("element").Position(p =>
p.X("Right").Y("Bottom").Width("300").Height("150").Buttons(item =>
 {
 item.ModelValue(ViewBag.ToastButtons1).Add();
 item.ModelValue(ViewBag.ToastButtons2).Add();
 }).ContentTemplate(@<div>
 <p>

 Anjolie Stokes
 </p>
 <div class="content">
 <p>Thanks for update!</p>
 </div>
 </div>).Render()
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 var value =
parseInt(document.getElementById('toast_input_index').value)
 toastObj.show({ timeOut: value });
 });
</script>
<style>
 .toast-img {
 width: 40px;
 height: 40px;
 }
 .name {
 padding-left: 20px;
 font-size: 17px;
 }
</style>
```

```

 font-weight: 500;
 }
 .content {
 padding-left: 60px;
 font-size: 12px;
 }
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.ToastButtons1 = new DefaultButtonModel() { content =
"Ignore" };
 ViewBag.ToastButtons2 = new DefaultButtonModel() { content = "reply"
};
 return View();
 }
 public class DefaultButtonModel
 {
 public string content { get; set; }
 }
}

```

## Static toast

You can prevent auto hiding in a toast as visible like static by setting zero (0) value in the [TimeOut](#) Property.

## CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to
accept").Timeout(0).ShowCloseButton(true).Position(p =>
p.X("Right")).Render()
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
</script>

```

**CONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

See Also

- [How to close the toast with click/tab](#)

**Template**

The [Template](#) property can be given as the **HTML element**; this can be either a **string** or **query selector**.

The HTML element tag can be given as a string for the Template property.

```
`typescript
```

```
template: "<div>Toast Content</div>"
```

```
,
```

The Template property allows getting the template content using the **query selector**. Here, the element 'ID' attribute is specified in the template.

```
`typescript
```

```
template: "#Template"
```

```
,
```

**CSHTML**

```
<div class="control-section" style="width:400px;margin:0 auto;">
 <div id="template_toast">

@Html.EJS().Toast("element").Timeout(120000).ExtendedTimeout(0).Position(p
=> p.X("Right").Y("Bottom")).Render()
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>

 <div id='result'></div>
</div>
<script id="template_toast_ele" type="text/x-template">
 <div id='template_toast' style="display: none">
 <div class="horizontal-align">
 <div class='toast-content'>
 <div class='toast-title'>
 Weekend Alarm
 </div>
 <div class='toast-message'>
```

```

 With traffic, its likely to take 45 minutes to get to
jenny's 24th Birthday Bash at Hillside Bar, 454 E.
 Olive Way by 10:00PM
 </div>
</div>
</div>

</div>
</script>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show({ template:
document.getElementById('template_toast_ele').innerHTML });
 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
</script>
<style>
 #snooze,
 #dismiss {
 background-color: #fff;
 }
 body > #element .e-toast {
 width: 400px !important;
 }
 .toast-template-section #reminder {
 text-align: center;
 margin: 15px;
 }
 #toast_custom .e-toast-template {
 display: inline-flex;
 }
 #template_toast .toast-icons {
 font-size: 35px;
 height: auto;
 margin: auto;
 }
 #template_toast .toast-icons.e-alarm::before {
 content: "\e702";
 }
 #template_toast .horizontal-align {
 display: inline-flex;
 flex-direction: row;
 width: 100%;
 }
 #template_toast .horizontal-align,
 #template_toast #snoozedropDown,
 #template_toast .snooze,

```

```

#template_toast .snoozeBtn {
 margin: 10px 0;
}
#template_toast .horizontal-align .toast-content .toast-title {
 font-weight: 500;
}
#template_toast .horizontal-align .toast-content .toast-message {
 opacity: 0.4;
}
#template_toast .horizontal-align .toast-content {
 display: inline-flex;
 flex: 1;
 flex-direction: column;
 margin-left: 10px;
}
</style>

```

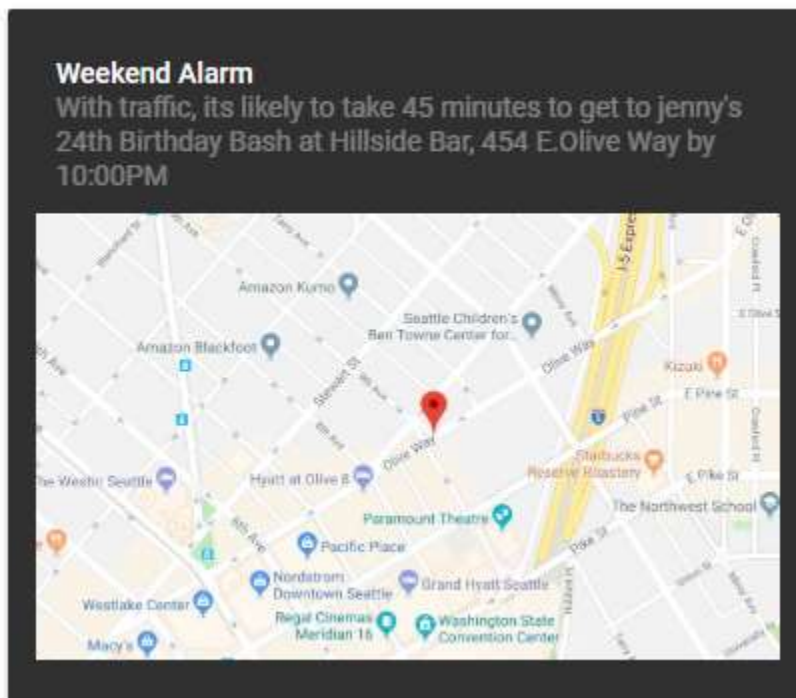
### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

Output be like the below.





## See Also

- [How to add dynamic template](#)
- [How to play an audio before open the toast](#)

## Animations

The toast control supports custom animations for both shows and hide actions from the provided [Animation](#) option of the `Animation` library.

The default animation is given as `FadeIn` for showing the toast and `FadeOut` for hiding the toast.

The following sample demonstrates some types of animations that suit toast. You can check all the animation effects here.

**CSHTML**

```
<div id="default" style="padding-bottom:75px;">
 <div class='row'>
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
 <div class='row'>
 <div class="col-md-6">
 <label> Show Animation </label>
 </div>
 <div class="col-md-6">
 @Html.EJS().DropDownList("showAnimation").Fields(e =>
e.Text("Value").Value("Id")).Placeholder("Select a animate
type").PopupHeight("200px").Index(0).DataSource((IEnumerable<object>)ViewBag
.data).Change("valueChange").Render()
 </div>
 </div>
 <div class='row'>
 <div class="col-md-6">
 <label> Hide Animation </label>
 </div>
 <div class="col-md-6">
 @Html.EJS().DropDownList("hideAnimation").Fields(e =>
e.Text("Value").Value("Id")).Placeholder("Select a animate
type").PopupHeight("200px").Index(0).DataSource((IEnumerable<object>)ViewBag
.data).Change("valueChange1").Render()
 </div>
 </div>
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to accept").Position(p
=> p.X("Right").Y("Bottom")).Render()
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.animation.show.effect = "FadeIn";
 toastObj.animation.hide.effect = "FadeOut";
 toastObj.show();
 }
);
</script>
```

```

 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
});
function valueChange(e) {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.animation.show.effect = e.value;
}
function valueChange1(e) {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.animation.hide.effect = e.value;
}
</script>
<style>
 #default {
 width: 600px;
 margin: 0 auto;
 }
 .row {
 margin: 15px;
 }
</style>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.data = new Animation().Animations();
 return View();
 }
 public class Animation
 {
 public string Value { get; set; }
 public string Id { get; set; }
 public List<Animation> Animations()
 {
 List<Animation> animation = new List<Animation>
 {
 new Animation { Id = "FadeIn", Value = "Fade In" },
 new Animation { Id = "FadeZoomIn", Value = "Fade Zoom
In" },
 new Animation { Id = "FadeZoomOut", Value = "Fade Zoom
Out" },
 new Animation { Id = "FlipLeftDownIn", Value = "Flip
Left Down In" },
 new Animation { Id = "FlipLeftDownOut", Value = "Flip
Left Down Out" },
 new Animation { Id = "FlipLeftUpIn", Value = "Flip Left
Up In" },
 new Animation { Id = "FlipLeftUpOut", Value = "Flip Left
Up Out" },
 }
 }
 }
}

```

```

Right Down In" },
Right Up In" },
Right Up Out" },
Bottom In" },
Bottom Out" },
In" },
Out" },
Right In" },
Right Out" },
In" },
Out" },

 new Animation { Id = "ZoomIn", Value = "Zoom In" },
 new Animation { Id = "ZoomOut", Value = "Zoom Out" }
};
return animation;
}
}
}

```

## Toast Utility Services

The Toast component provides a built-in utility function to render the toast with minimal code. The utility function will render the toast without the need of rendering the container element in the DOM where the toast is appended. So that, the toast can now be rendered on the go. The following are the option to render the toast using the utility function.

### Show Toast with predefined types

The Toast component support 4 types of predefined toast with essential colors for various situations which can be shown using the `ToastUtility.show` by just defining the type of the toast without defining any class names. The following options are used as an argument on calling the utility function for predefined types:

| Options              | Description                                                                                                                                                                                                                  |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>content</code> | Specifies the content that can be displayed on the Toast.                                                                                                                                                                    |
| <code>type</code>    | Specifies the type of the predefined Toasts. The 4 types of predefined toasts are <code>Information</code> , <code>Success</code> , <code>Error</code> , <code>Warning</code>                                                |
| <code>timeOut</code> | Specifies the Toast display time duration on the page in milliseconds. Once the time expires, Toast message will be removed. Setting 0 as a time out value displays the Toast on the page until the user closes it manually. |

**CSHTML**

```

<div class="control-section" style="width:400px;margin:0 auto;">
 <div class="row">
 @Html.EJS().Button("info_toast").Content("Info Toast").CssClass("e-
 btn e-control e-info").Render()
 @Html.EJS().Button("success_toast").Content("Success
 Toast").CssClass("e-btn e-control e-success").Render()
 @Html.EJS().Button("warning_toast").Content("Warning
 Toast").CssClass("e-btn e-control e-warning").Render()
 @Html.EJS().Button("danger_toast").Content("Danger
 Toast").CssClass("e-btn e-control e-danger").Render()
 </div>

 <div class="row" style="text-align: center;">
 @Html.EJS().Button("hide_toast").Content("Hide All").CssClass("e-
 btn").Render()
 </div>
</div>
<script type="text/javascript">
 var toastObj;
 document.getElementById("info_toast").addEventListener('click', function
 () {
 toastObj = ej.notifications.ToastUtility.show('Please read the
 comments carefully', 'Information', 20000);
 });
 document.getElementById("success_toast").addEventListener('click',
 function () {
 toastObj = ej.notifications.ToastUtility.show('Your message has been
 sent successfully', 'Success', 20000);
 });
 document.getElementById("warning_toast").addEventListener('click',
 function () {
 toastObj = ej.notifications.ToastUtility.show('There was a problem
 with your network connection', 'Warning', 20000);
 });
 document.getElementById("danger_toast").addEventListener('click',
 function () {
 toastObj = ej.notifications.ToastUtility.show('A problem has been
 occurred while submitting the data', 'Error', 20000);
 });
 document.getElementById("hide_toast").addEventListener('click', function
 () {
 toastObj.hide('All');
 });
</script>

```

**CONTROLLER.CS**

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Show Toast with ToastModel

The utility function can be called using the [ToastModel](#) as argument to show the toast where all the properties in the `ToastModel` like any events, position, close icon, action buttons, etc. can be used in the `ToastUtility.show`.

#### CSHTML

```
<div class="control-section" style="width:400px;margin:0 auto;">
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
 btn").Render()
 </div>
</div>
<script type="text/javascript">
 var toastObj;
 document.getElementById("show_toast").addEventListener('click', function
 () {
 toastObj = ejs.notifications.ToastUtility.show({
 title: 'Toast Title',
 content: 'Toast is shown using the utility function with
 ToastModel',
 timeOut: 20000,
 position: { X: 'Right', Y: 'Bottom' },
 showCloseButton: true,
 click: toastClick,
 buttons: [{
 model: { content: 'Close' }, click: toastClose
 }]
 });

 function toastClick() {
 console.log('Toast click event triggered');
 }
 function toastClose() {
 toastObj.hide();
 }
 });
</script>
```

#### CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

### CSS structures

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

### Customizing the toast title

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

```
`CSS
/ To change color, font family and font size /
.e-toast-container .e-toast .e-toast-message .e-toast-title {
color: red;
font-size: 18px;
font-weight: bold;
}
`
```

### Customizing the toast content

Use the following CSS to customize the default toast's content properties like font-family, font-size and color.

```
`CSS
/ To change color, font family and font size /
.e-toast-container .e-toast .e-toast-message .e-toast-content {
color: aqua;
font-size: 13px;
font-weight: normal;
}
`
```

### Customizing the toast icon

Use the following CSS to customize the default toast icon color.

```
`CSS
/ To change icon color /
.e-toast-container .e-toast .e-toast-icon {
color: yellow;
}
`
```

### Customizing the toast background

Use the following CSS to customize the default toast's background color.

```
`CSS
/ To change background color /
.e-toast-container .e-toast {
```

```
background-color: navy;
}
`
```

## Accessibility

The Toast control has been designed with [WAI-ARIA](#) specifications in mind by applying the prompt WAI-ARIA roles, states, and properties with the keyboard support. It helps users who use assistive WAI-ARIA accessibility support, which is achieved using attributes.

It provides information about the elements in a document for assistive technology.

The Toast control implements the keyboard navigation support by using the following [WAI-ARIA practices](#) and is tested in major screen readers.

The Toast component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Toast component is outlined below.

```
| Accessibility Criteria | Compatibility |
| -- | -- |
| WCAG 2.2 Support | |
| Section 508 Support | |
| Screen Reader Support | |
| Right-To-Left Support | |
| Color Contrast | |
| Mobile Device Support | |
| Keyboard Navigation Support | |
| Accessibility Checker Validation | |
| Axe-core Accessibility Validation | |
<style>
.post .post-content img {
display: inline-block;
margin: 0.5em 0;
```

```

}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

## WAI-ARIA attributes

```
<!-- markdownlint-disable MD033 -->
```

| Class | Description |
|-------|-------------|
|-------|-------------|

\_\_\_\_\_

| role | **alert:**   
      Identifies the element as a container when alert content will be added or updated. |

## CSHTML

```
<div class="control-section">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to accept").Render()
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
</script>
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

## Ensuring accessibility

The Toast component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Toast component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Toast component with accessibility tools.



See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## How To

### Prevent duplicate toast display

You can prevent identical same toast displaying in a screen by the event function and terminate the toast displaying process by setting the cancel event property in the [BeforeOpen](#) event.

The following sample demonstrates preventing duplicate title toast element displaying with custom code blocks.

### CSHTML

```
<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Sample Toast Title").Position(p =>
 p.X("Center")).Content("Sample Toast
 content").BeforeOpen("onBeforeOpen").Close("onClose").Created("onCreate").Re
 nder()
 @Html.EJS().Button("showToast").Content("Show Toast").CssClass("e-
 btn").Render()
</div>
<script type="text/javascript">
 var toasts = [
 { title: 'Warning !', content: 'There was a problem with your
 network connection.', isOpen: false },
 { title: 'Success !', content: 'Your message has been sent
 successfully.', isOpen: false },
 { title: 'Error !', content: 'A problem has been occurred while
 submitting your data.', isOpen: false }
];
 var toastFlag = 0;
 setTimeout(
 () => {
 var toastObj =
 document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 }, 1000);
 function onBeforeOpen(e) {
 if (preventDuplicate(e)) {
 e.cancel = true;
 }
 }
 function preventDuplicate(e) {
 var toastEle = e.element;
 var toasts = e.toastObj.element.children;
 for (var i = 0; i < toasts.length; i++) {
 if (toasts[i].title === e.options.title && !toasts[i].isOpen) {
 toasts[i].isOpen = true;
 return false;
 }
 }
 return true;
 }
</script>
```

```

 }
 function onCreate() {
 toasts.show(toasts[toastFlag]);
 ++toastFlag;
 }
 function onClose(e) {
 for (let i: number = 0; i < toasts.length; i++) {
 if (toasts[i].title === e.options.title) {
 toasts[i].isOpen = false;
 }
 }
 }
 document.getElementById("showToast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
 });
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Restrict the maximum toast to show

You can restrict the maximum toast count by using the event callback function and terminate the toast displaying process by setting the cancel event property in the [BeforeOpen](#) event.

The following sample demonstrates restricting toast displaying up to 3. You can restrict by your own count with custom code blocks.

### CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Sample Toast Title").Position(p =>
p.X("Center")).Content("Sample Toast
content").BeforeOpen("onBeforeOpen").Render()
 @Html.EJS().Button("showToast").Content("Show Toast").CssClass("e-
btn").Render()
</div>
<script type="text/javascript">
 var toasts = [
 { title: 'Warning !', content: 'There was a problem with your
network connection.' },
 { title: 'Success !', content: 'Your message has been sent
successfully.' },
]

```

```

 { title: 'Error !', content: 'A problem has been occurred while
submitting your data.' },
 { title: 'Information !', content: 'Please read the comments
carefully.' }
];
 var maxCount = 3;
 var toastFlag = 0;
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 }, 1000);
 function onBeforeOpen(e) {
 var toastObj = document.getElementById('element').ej2_instances[0];
 if (maxCount === toastObj.element.childElementCount) {
 e.cancel = true;
 } else {
 e.cancel = false;
 }
 }
 document.getElementById("showToast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
 });
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

## Customize progress bar theme and sizing

By default, the progress bar appears based on the theme stylings and dimensions. You can customize the progress bar stylings using custom CSS or event functions.

The following sample demonstrates customizing the progress bar stylings using the [BeforeOpen](#) event.

## CSHTML

```

<div style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Position(p => p.X("Right").Y("Bottom")).Content("You have a new

```

```

friend request yet to
accept").BeforeOpen("onBeforeOpen").ShowProgressBar(true).Render()
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
 <div class="row" style="padding-top: 20px">
 <div class="e-float-input">
 <input class="e-input" id="progressHeight" name="Digits"
value="4" required>

 <label class="e-float-text" for="Digits">Progress Bar
Height</label>
 </div>
 </div>
 <div class="row" style="padding-top: 20px">
 <div class="col-md-6">
 <label> Progress Bar Color </label>
 </div>
 <div class="col-md-6">
 @Html.EJS().DropDownList("Progress").Placeholder("Select a
animate type").PopupHeight("150px").Index(0).DataSource(
(IEnumerable<object>)ViewBag.data).Change("valueChange").Fields(e=>e.Value("
Id").Text("Value")).Render()
 </div>
 </div>
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 200);
 function onBeforeOpen(e) {
 var progress = e.element.querySelector('.e-toast-progress');
 progress.style.height =
document.getElementById('progressHeight').value + 'px';
 var listObjprogressColor =
document.getElementById('Progress').ej2_instances[0];
 progress.style.backgroundColor = listObjprogressColor.value;
 }
 function valueChange(e) {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 var listObjprogressColor =
document.getElementById('Progress').ej2_instances[0];
 var progressEles = toastObj.element.querySelectorAll('.e-toast-
progress');
 progressEles.forEach((ele) => {
 ele.style.backgroundColor = listObjprogressColor.value;
 })
 }
 document.getElementById("show_toast").addEventListener('click',
function () {

```

```

 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 ViewBag.data = new Color().Colors();
 return View();
 }
 public class Color
 {
 public string Value { get; set; }
 public string Id { get; set; }
 public List<Color> Colors()
 {
 List<Color> color = new List<Color>();
 position.Add(new Color { Id = "red", Value = "Red" });
 position.Add(new Color { Id = "cyan", Value = "Cyan" });
 position.Add(new Color { Id = "blue", Value = "Blue" });
 position.Add(new Color { Id = "yellow", Value = "Yellow" });
 position.Add(new Color { Id = "pink", Value = "Pink" });
 return color;
 }
 }
}

```

## Play an audio before open the toast

The following sample demonstrates how to play an audio in background while opening the toast by including audio play codes into the [BeforeOpen](#) event function.

**Note:** To stop the audio after displaying the toast, use the [Open](#) event in toast. For further customization, check the Toast Events APIs.

## CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Position(p => p.X("Right").Y("Bottom")).Content("You have a new
friend request yet to accept").BeforeOpen("onBeforeOpen").Render()
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
</div>

<script type="text/javascript">
 setTimeout(
 () => {

```

```

 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 function onBeforeOpen(e) {
 let audio = new
Audio('https://drive.google.com/uc?export=download&id=1M95VOptolcQ4FQHzNBaLf
0WFQglrtWi7');
 audio.play();
 }
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Show different types of toast

The Syncfusion ASP.NET Core toast has the following predefined styles that can be defined using the [CssClass](#) property for achieving different types of toast:

| Class           | Description                     |
|-----------------|---------------------------------|
| -----           | -----                           |
| e-toast-success | Represents a positive toast     |
| e-toast-info    | Represents an informative toast |
| e-toast-warning | Represents a toast with caution |
| e-toast-danger  | Represents a negative toast     |

### CSHTML

```

<div style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Position(p =>
p.X("Right").Y("Bottom")).Render()
 <div class="row">
 @Html.EJS().Button("showToast").Content("Show Types").CssClass("e-
btn").Render()
 </div>
</div>
<script type="text/javascript">
 var toasts = [

```

```

 { title: 'Warning !', content: 'There was a problem with your
network connection.', cssClass: 'e-toast-warning' },
 { title: 'Success !', content: 'Your message has been sent
successfully.', cssClass: 'e-toast-success' },
 { title: 'Error !', content: 'A problem has been occurred while
submitting your data.', cssClass: 'e-toast-danger' },
 { title: 'Information !', content: 'Please read the comments
carefully.', cssClass: 'e-toast-info' }]];
 var toastFlag = 0;
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 }, 200);
 document.getElementById("showToast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
 });
</script>

```

## CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Show multiple toasts in various positions

By default, the positions of the new toasts are only updated after the visible toasts have been destroyed. If You need to display multiple toasts with different positions, initiate another toasts.

The following sample demonstrates adding multiple toasts in different positions.

## CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Position(p =>
p.X("Right").Y("Bottom").Title("Warning !").Content("There was a problem
with your network connection.").Click("onClick").Render()
 @Html.EJS().Toast("element1").Position(p =>
p.X("Left").Y("Bottom").Title("Success !").Content("Your message has been
sent successfully.").Click("onClick").Render()
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Right Position
Toast").CssClass("e-btn").Render()
 </div>
</div>

```

```

 @Html.EJS().Button("show_toast1").Content("Show Left Position
Toast").CssClass("e-btn").Render()
 </div>
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 var toastObj =
document.getElementById('element1').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 200);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
 document.getElementById("show_toast1").addEventListener('click',
function () {
 var toastObj = document.getElementById('element1').ej2_instances[0];
 toastObj.show();
 });
 function onClick(e) {
 e.clickToClose = true;
 }
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Close the toast with click/tap

By default, the toasts are expired based on the [TimeOut](#) value. You can customize the toast hiding process with [click/tap](#) action by setting the event args in the [clicked](#) callback function with [static Toast](#).

### CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Position(p => p.X("Right").Y("Bottom")).Content("You have a new
friend request yet to accept").TimeOut(0).Click("onClick").Render()
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
</div>

```



```

</div>
</div>
<script type="text/javascript">
 setTimeout(
 () => {
 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 200);
 document.getElementById("show_toast").addEventListener('click', function
() {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show();
 });
 function onClick(e) {
 e.clickToClose = true;
 }
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Add dynamic template

Toast supports to change templates dynamically with displaying in multiple toasts. You can change the toast properties while calling in the `show` method.

### CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Position(p =>
p.X("Right").Y("Bottom")).Click("onClick").Render()
 <div id='templateToast' style="display: none;color:red"> System affected
by virus !!! </div>
 <div class="row">
 @Html.EJS().Button("show_toast").Content("Show Toast").CssClass("e-
btn").Render()
 </div>
</div>
<script type="text/javascript">
 var toastFlag = 0;
 var toasts = [{ template: '2 Mail has received' },
 { template: 'User Guest Logged in' },
 { template: 'Logging in as Guest' },
 { template: 'Ticket has reserved' },
 { template: '#templateToast' }];
 setTimeout(
 () => {

```

```

 var toastObj =
document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 }, 1000);
 document.getElementById("show_toast").addEventListener('click', function
 () {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.show(toasts[toastFlag]);
 ++toastFlag;
 if (toastFlag === (toasts.length)) {
 toastFlag = 0;
 }
 });
 function onClick(e) {
 e.clickToClose = true;
 }
</script>

```

### CONTROLLER.CS

```

public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}

```

### Prevent toast close with mobile swipe

You can prevent the toast close with mobile swipe action by setting [BeforeClose](#) argument cancel value to true while argument type as a swipe. The following code shows how to prevent toast close with mobile swipe.

The following sample demonstrates preventing toast close with mobile swipe element displaying with custom code blocks.

### CSHTML

```

<div class="control-section" style="width:400px;margin:0 auto;">
 @Html.EJS().Toast("element").Title("Matt sent you a friend
request").Content("You have a new friend request yet to
accept").BeforeClose("beforeClose").Render()
 @Html.EJS().Button("button").Content("Show Toast").CssClass("e-
btn").Render()
</div>
<script type="text/javascript">
 setTimeout(() => {
 var toastObj = document.getElementById('element').ej2_instances[0];
 toastObj.target = document.body;
 toastObj.show();
 }, 1000);
 document.getElementById("button").addEventListener('click', function ()
 {

```

```
var toastObj = document.getElementById('element').ej2_instances[0];
toastObj.show();
});
function beforeClose(args: ToastBeforeCloseArgs) {
 if (args.type === "swipe") {
 args.cancel = true;
 }
}
</script>
```

## CONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View();
 }
}
```

## ToolBar

### Getting Started with ASP.NET MVC Toolbar Control

This section briefly explains about how to include [ASP.NET MVC Toolbar](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

## PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Toolbar control

Now, add the Syncfusion ASP.NET MVC Toolbar control in **~/Views/Home/Index.cshtml** page.

#### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().Toolbar("defaultToolbar")
.Width("600px")
.Items(new List<ToolBarItem> {
```

```

 new ToolbarItem { Text="Cut", PrefixIcon = "e-cut-icon tb-icons",
 TooltipText = "Cut" },
 new ToolbarItem { Text="Copy", PrefixIcon = "e-copy-icon tb-icons",
 TooltipText = "Copy" },
 new ToolbarItem { Text="Paste", PrefixIcon = "e-paste-icon tb-
icons", TooltipText = "Paste" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Text="Bold", PrefixIcon = "e-bold-icon tb-icons",
 TooltipText = "Bold" },
 new ToolbarItem { Text="Underline", PrefixIcon = "e-underline-icon
tb-icons", TooltipText = "Underline" },
 new ToolbarItem { Text="Italic", PrefixIcon = "e-italic-icon tb-
icons", TooltipText = "Italic" },
 new ToolbarItem { Text="Color Picker", PrefixIcon = "e-color-icon
tb-icons", TooltipText = "Color-Picker" }
 })
 .Render()
)

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Toolbar control will be rendered in the default web browser.

Cut Copy Paste Bold Underline Italic Color Picker

### Render the Toolbar items using content template

You can bind any HTML elements or other controls in Toolbar items, by simply using the content template property in ASP.NET Toolbar.

In the below demo, the Toolbar items are given as [Button](#), [MaskedTextBox](#), [RadioButton](#), [DropDownList](#) using the content template. In the content template property of Toolbar, you can directly render these controls like below in the code.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
@using Syncfusion.EJ2.DropDowns;
@model List<string>
@ (Html.EJS().Toolbar("defaultToolbar")
 .ContentTemplate(
 @<div>
 <div>

<div>@Html.EJS().Button("btn").Content("Click").IsPrimary(true).Render() </di
v>

 <div class='separator'> </div>
 <div>@Html.EJS().MaskedTextBox("mask1").Mask("345-678-
4673").Render() </div>
 <div class='separator'> </div>
 <div>@Html.EJS().RadioButton("radio1").Label("Credit/Debit
Card").Name("payment").Value("credit/debit").Render() </div>
 <div class='separator'> </div>
 <div>
 @Html.EJS().DropDownList("games").Placeholder("Select a
game").PopupHeight("220px").Index(2).DataSource(

```

```

 (IEnumerable<object>)Model).Fields(new
 DropDownListFieldSettings { Text = "Game", Value = "Id" }).Render()
 }
}
</div>
</div>
</div>
)
.Render()
)
<style>
 .separator {
 width: 10px;
 }
</style>

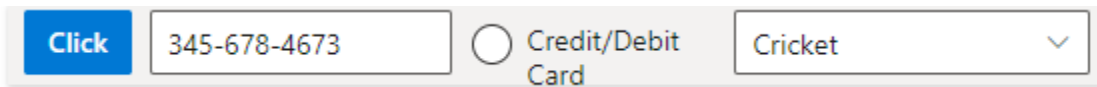
```

### HOMECONTROLLER.CS

```

public ActionResult Index()
{
 List<string> data = new List<string>() { "Badminton", "Basketball",
 "Cricket", "Football", "Golf", "Gymnastics", "Hockey", "Tennis" };
 return View(data);
}

```



**Note:** [View Sample in GitHub.](#)

See also

- [How to add Toggle Button](#)

### Item Configuration in ASP.NET MVC Toolbar control

The Toolbar can be rendered by defining an array of items. Items can be constructed with the following built-in command types or item template.

#### Button

**Button** is the default command type, and it can be rendered by using the **text** property. Properties of the button command type:

| Property   | Description                                                                                                                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| text       | The text to be displayed for button.                                                                                                                                                                  |
| id         | The ID of the button to be rendered. If the ID is not given, auto ID is generated.                                                                                                                    |
| prefixIcon | Defines the class used to specify an icon for the button. The icon is positioned before the text if text is available or the icon alone button is rendered.                                           |
| suffixIcon | Defines the class used to specify an icon for the button. The icon is positioned after the text if text is available. If both prefixIcon and suffixIcon are specified, only prefixIcon is considered. |

| width | Used to set the width of the button. |

### Separator

The **Separator** type adds a vertical separation between the Toolbar's single/multiple commands.

### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().ToolBar("defaultToolBar")
 .Width("300px")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Text = "Cut" },
 new ToolBarItem { Text = "Copy" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Text = "Paste" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Text = "Undo" },
 new ToolBarItem { Text = "Redo" }
 })
 .Render()
)
```

### SEPARATOR.CS

```
public ActionResult Index()
{
 return View();
}
```

**Note:** If **Separator** is added as the first or the last item, it will not be visible.

### Input

The **Input** type is only applicable for adding **template** elements when the **template** property is defined as an **object**. Input type creates an **input element** internally that acts as the container for **Syncfusion** input based components.

Note: Set toolbar item **type** property value as **Input** only for Input components.

### NumericTextBox

- The **NumericTextBox** component can be included by importing the **NumericTextBox** module from ej2-inputs.
- Initialize the **NumericTextBox** in template property, where the Toolbar item type is set as **Input**.
- Related **NumericTextBox** component properties can also be configured as given below.

```
`javascript
```

```
<ejs-numerictextbox format="n2"></ejs-numerictextbox>
```

```
,
```

*DropDownList*

- The **DropDownList** component can be included by importing the **DropDownList** module from **ej2-dropdowns**.
- Initialize the **DropDownList** in template property, where the Toolbar item type is set as **Input**.
- Related **DropDownList** component properties can also be configured as given below.

```
`javascript
```

```
<ejs-dropdownlist width="100"></ejs-dropdownlist>
```

```
,
```

*RadioButton*

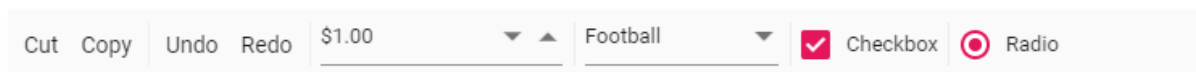
- The **RadioButton** component can be included by importing the **RadioButton** module from **ej2-buttons**.
- Initialize the **RadioButton** in template property, where the Toolbar item type is set as **Input**.
- Related **RadioButton** component properties can also be configured as given below.

```
`javascript
```

```
<ejs-radiobutton label="Option 1" name="default"></ejs-radiobutton>
```

```
,
```

Output be like the below.

*Enabling tab key navigation in Toolbar*

The **tabIndex** property of a Toolbar item is used to enable tab key navigation for the item. By default, the user can switch between items using the arrow keys, but the **tabIndex** property allows you to switch between items using the Tab and Shift+Tab keys as well.

To use the **tabIndex** property, you need to set it for each Toolbar item that you want to enable tab key navigation. The **tabIndex** property should be set to a positive integer value. A value of 0 or a negative value will disable tab key navigation for the item.

For example, to enable tab key navigation for two Toolbar items, you can use the following code:

```
`javascript
```

```
@using Syncfusion.EJ2.Navigations;
```

```
<ejs-toolbar id="defaultToolbar">
```

```
<e-toolbar-items>
```

```
<e-toolbar-item text="Item 1" tabIndex = "1"></e-toolbar-item>
```

```
<e-toolbar-item text="Item 2" tabIndex = "2"></e-toolbar-item>
```



```
</e-toolbar-items>
```

```
</ejs-toolbar>
```

```
,
```

With the above code, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, in addition to using the arrow keys. The items will be navigated in the order specified by the `tabIndex` values.

If you set the `tabIndex` value to 0 for all Toolbar items, tab key navigation will be based on the element order rather than the `tabIndex` values. For example:

```
`javascript
```

```
@using Syncfusion.EJ2.Navigations;
```

```
<ejs-toolbar id="defaultToolbar">
```

```
<e-toolbar-items>
```

```
<e-toolbar-item text="Item 1" tabIndex = "0"></e-toolbar-item>
```

```
<e-toolbar-item text="Item 2" tabIndex = "0"></e-toolbar-item>
```

```
</e-toolbar-items>
```

```
</ejs-toolbar>
```

```
,
```

In this case, the user can switch between the two Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated in the order in which they appear in the DOM.

Example:

Here is an example of how you can use the `tabIndex` property to enable tab key navigation for a Toolbar component:

### **CSSHTML**

```
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().Toolbar("defaultToolbar")
 .Width("300px")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Text = "Cut", TabIndex = 0 },
 new ToolBarItem { Text = "Copy", TabIndex = 0 },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Text = "Paste", TabIndex = 0 },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Text = "Undo", TabIndex = 0 },
 new ToolBarItem { Text = "Redo", TabIndex = 0 }
 })
 .Render()
)
```

### **TABKEYNAVIGATION.CS**

```
public ActionResult Index()
{
```

```
return View();
}
```

With the above code, the user can switch between the Toolbar items using the Tab and Shift+Tab keys, and the items will be navigated based on the element order.

See Also

- [How to set item wise custom template](#)

## Responsive Mode

This section explains the supported display modes of the Toolbar when the content exceeds the viewing area. Possible modes are:

- Scrollable
- Popup

## Scrollable

The default overflow mode of the Toolbar is **Scrollable**. Scrollable display mode supports display of commands in a single line with horizontal scrolling enabled when commands overflow to available space.

- The right and left navigation arrows are added to the start and end of the Toolbar to navigate to hidden commands.
- You can also see the hidden commands using touch swipe action.
- By default, if navigation icon in the **left** side is disabled, you can see the hidden commands by moving to the **right**.
- By clicking the arrow or by holding the arrow continuously, hidden commands will become visible.
- If device navigation icons are not available, you can touch swipe to see the hidden commands of the Toolbar.



- Once the Toolbar reaches the last or first command, the corresponding navigation icon will be disabled and you can move to the opposite direction.



- You can continuously scroll the Toolbar content by holding the navigation icon.



### CSHTML

```
@using Syncfusion.EJ2.Navigations;
@(Html.EJS().ToolBar("defaultToolBar")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-cut-icon",
 Text = "Cut" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-copy-
 icon", Text = "Copy" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-paste-
 icon", Text = "Paste" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-bold-
 icon", Text = "Bold" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-underline-
 icon", Text = "Underline" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-italic-
 icon", Text = "Italic" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-color-
 icon", Text = "Color-Picker" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-ascending-
 icon", Text = "A-Z Sort" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-
 descending-icon", Text = "Z-A Sort" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-clear-
 icon", Text = "Clear" },
 })
 .Width(1000)
 .Height(40)
 .Render())
```

```
 })
 .Width("600")
 .Render()
)
<style>
/* Toolbar Styles */
.e-cut-icon:before {
 content: "\e604"
}
.e-copy-icon:before {
 content: "\e70a"
}
.e-paste-icon:before {
 content: "\e712"
}
.e-color-icon:before {
 content: "\e703";
}
.e-bold-icon:before {
 content: "\e339"
}
.e-underline-icon:before {
 content: "\e706";
}
.e-alignleft-icon:before {
 content: "\e717"
}
.e-alignright-icon:before {
 content: "\e715"
}
.e-aligncenter-icon:before {
 content: "\e704"
}
.e-alignjustify-icon:before {
 content: "\e71b"
}
.e-upload-icon:before {
 content: "\e71e"
}
.e-download-icon:before {
 content: "\e70a"
}
.e-indent-icon:before {
 content: "\e70b"
}
.e-outdent-icon:before {
 content: "\e700"
}
.e-clear-icon:before {
 content: "\e70d"
}
.e-reload-icon:before {
 content: "\e71c"
}
.e-export-icon:before {
 content: "\e720";
}
}
```

```

.e-italic-icon:before {
 content: "\e710"
}
.e-bullets-icon:before {
 content: "\e711";
}
.e-numbering-icon:before {
 content: "\e70e";
}
.e-ascending-icon:before {
 content: "\e70f";
}
.e-descending-icon:before {
 content: "\e707";
}
}
</style>

```

### SCROLLABLE.CS

```

public ActionResult Index()
{
 return View();
}

```

### Popup

**Popup** is another type of **overflowMode** in which the Toolbar container holds the commands that can be placed in the available space. The rest of the overflowing commands that do not fit within the viewing area moves to the overflow popup container.

The commands placed in the popup can be viewed by opening the popup using the drop down icon given at the end of the Toolbar.



**Note:** If the popup content overflows the height of the page, then the rest of the commands will be hidden.

### Priority of commands

Default popup priority is set as **none**, and when the commands of the Toolbar overflow, the ones listed last will be moved to the popup.

You can customize the priority of commands to be displayed on the Toolbar and popup by using the **Overflow** property.

Property | Description

Both | Button text is visible in both **ToolBar** and **Popup**.

Overflow | Button text is only visible in **Popup**.

ToolBar | Button text is only visible on the **ToolBar**.

In the following code sample, text is only visible in the popup container and not in the Toolbar container.

### **CSHTML**

```
@using Syncfusion.EJ2.Navigations;
@ (Html.EJS().ToolBar("defaultToolBar")
 .Items(ViewBag.popItems)
 .Width("330")
 .OverflowMode(OverflowMode.Popup)
 .Render()
)
<style>
 /* Toolbar Styles */
 .e-cut-icon:before {
 content: "\e604"
 }
 .e-copy-icon:before {
 content: "\e70a"
 }
 .e-paste-icon:before {
 content: "\e712"
 }
 .e-color-icon:before {
 content: "\e703";
 }
 .e-bold-icon:before {
 content: "\e339"
 }
 .e-underline-icon:before {
 content: "\e706";
 }
 .e-alignleft-icon:before {
 content: "\e717"
 }
 .e-alignright-icon:before {
 content: "\e715"
 }
 .e-aligncenter-icon:before {
 content: "\e704"
 }
 .e-alignjustify-icon:before {
 content: "\e71b"
 }
 .e-upload-icon:before {
 content: "\e71e"
 }
}
```

```

.e-download-icon:before {
 content: "\e70a"
}
.e-indent-icon:before {
 content: "\e70b"
}
.e-outdent-icon:before {
 content: "\e700"
}
.e-clear-icon:before {
 content: "\e70d"
}
.e-reload-icon:before {
 content: "\e71c"
}
.e-export-icon:before {
 content: "\e720";
}
.e-italic-icon:before {
 content: "\e710"
}
.e-bullets-icon:before {
 content: "\e711";
}
.e-numbering-icon:before {
 content: "\e70e";
}
.e-ascending-icon:before {
 content: "\e70f";
}
.e-descending-icon:before {
 content: "\e707";
}
}
</style>

```

### TEXTBUTTON.CS

```

public ActionResult Index()
{
 List<ToolBarItem> popItems = new List<ToolBarItem>();
 popItems.Add(new ToolBarItem { PrefixIcon = "e-cut-icon", Text = "Cut",
 ShowTextOn = DisplayMode.Overflow, Overflow = OverflowOption.Show });
 popItems.Add(new ToolBarItem { PrefixIcon = "e-copy-icon", Text =
 "Copy", ShowTextOn = DisplayMode.Overflow, Overflow = OverflowOption.Show
 });
 popItems.Add(new ToolBarItem { PrefixIcon = "e-paste-icon", Text =
 "Paste", ShowTextOn = DisplayMode.Overflow, Overflow = OverflowOption.Show
 });
 popItems.Add(new ToolBarItem { Type = ItemType.Separator });
 popItems.Add(new ToolBarItem { PrefixIcon = "e-bold-icon", Text =
 "Bold", ShowTextOn = DisplayMode.Overflow, Overflow = OverflowOption.Show
 });
 popItems.Add(new ToolBarItem { PrefixIcon = "e-underline-icon", Text =
 "Underline", ShowTextOn = DisplayMode.Overflow, Overflow =
 OverflowOption.Show });
}

```

```

 popItems.Add(new ToolbarItem { PrefixIcon = "e-italic-icon", Text =
 "Italic", ShowTextOn = DisplayMode.Overflow, Overflow = OverflowOption.Show
 });
 popItems.Add(new ToolbarItem { PrefixIcon = "e-color-icon", Text =
 "Color-Picker", ShowTextOn = DisplayMode.Overflow, Overflow =
 OverflowOption.Show });
 popItems.Add(new ToolbarItem { Type = ItemType.Separator });
 popItems.Add(new ToolbarItem { PrefixIcon = "e-ascending-icon", Text =
 "A-Z Sort", ShowTextOn = DisplayMode.Overflow, Overflow =
 OverflowOption.Show });
 popItems.Add(new ToolbarItem { PrefixIcon = "e-descending-icon", Text =
 "Z-A Sort", ShowTextOn = DisplayMode.Overflow, Overflow =
 OverflowOption.Show });
 popItems.Add(new ToolbarItem { PrefixIcon = "e-clear-icon", Text =
 "Clear", ShowTextOn = DisplayMode.Overflow, Overflow = OverflowOption.Show
 });
 ViewBag.popItems = popItems;
 return View();
 }

```

## Template Configuration

The Toolbar can be rendered by item based collection and by HTML elements. To render it based on the given HTML element, use `id` as the `target` property. To render the Toolbar, follow the below structure of the HTML elements:

```
`html
```

```
<div id='template_toolbar'> --> Root Toolbar Element
```

```
<div> --> Toolbar Items Container
```

```
<div> --> Toolbar Item Element
```

```
</div>
```

```
</div>
```

```
</div>
```

```
,
```

Here, the template ID, `#template_toolbar` is directly appended to the Toolbar.

## CSHTML

```

<div id='template_toolbar'>
 <div>
 <div><button class='e-btn e-tbar-btn'>Cut</button> </div>
 <div><button class='e-btn e-tbar-btn'>Copy</button> </div>
 <div><button class='e-btn e-tbar-btn'>Paste</button> </div>
 <div class='e-separator'> </div>
 <div><button class='e-btn e-tbar-btn'>Bold</button> </div>
 <div><button class='e-btn e-tbar-btn'>Italic</button> </div>
 </div>
</div>
<script type="text/javascript">
 var toolbarObj = new ej.navigations.Toolbar({ });
 toolbarObj.appendTo('#template_toolbar');

```



```
</script>
```

## TEMPLATE.CS

```
public ActionResult Index()
{
 return View();
}
```

### Popup customization

**Popup** is one of the supported responsive modes of the Toolbar. The Toolbar commands, popup mode priority and button text mode customizations are achieved in the item based rendering through property declaration. For more information on popup mode, refer [here](#)

The above behavior can also be achieved with template rendering by defining equivalent class names instead of property declaration.

Equivalent class names listed below are needed to add the Toolbar items' **div** element.

#### Priority

| Class          | Description                                         |
|----------------|-----------------------------------------------------|
| e-popup-text   | Button text is only visible in the <b>Popup</b> .   |
| e-toolbar-text | Button text is only visible on the <b>Toolbar</b> . |

## CSHTML

```
<div id='template_toolbar'>
 <div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div class="e-
tbar-btn-text">Cut</div></button> </div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div class="e-
tbar-btn-text">Copy</div></button> </div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div
class="e-tbar-btn-text">Paste</div></button> </div>
 <div class='e-separator'> </div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div class="e-
tbar-btn-text">Bold</div></button> </div>
 <div class='e-overflow-hide e-popup-text'><button class='e-btn e-
tbar-btn'><div
class="e-tbar-btn-text">Underline</div></button> </div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div
class="e-tbar-btn-text">Italic</div></button> </div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div
class="e-tbar-btn-text">A-Z Sort</div></button> </div>
 <div class='e-overflow-show e-popup-text'><button class='e-btn e-
tbar-btn'><div
class="e-tbar-btn-text">Z-A Sort</div></button> </div>
 </div>
```

```
</div>
<style>
 /* Toolbar Styles */
 .e-cut-icon:before {
 content: "\e604"
 }
 .e-copy-icon:before {
 content: "\e70a"
 }
 .e-paste-icon:before {
 content: "\e712"
 }
 .e-color-icon:before {
 content: "\e703";
 }
 .e-bold-icon:before {
 content: "\e339"
 }
 .e-underline-icon:before {
 content: "\e706";
 }
 .e-alignleft-icon:before {
 content: "\e717"
 }
 .e-alignright-icon:before {
 content: "\e715"
 }
 .e-aligncenter-icon:before {
 content: "\e704"
 }
 .e-alignjustify-icon:before {
 content: "\e71b"
 }
 .e-upload-icon:before {
 content: "\e71e"
 }
 .e-download-icon:before {
 content: "\e70a"
 }
 .e-indent-icon:before {
 content: "\e70b"
 }
 .e-outdent-icon:before {
 content: "\e700"
 }
 .e-clear-icon:before {
 content: "\e70d"
 }
 .e-reload-icon:before {
 content: "\e71c"
 }
 .e-export-icon:before {
 content: "\e720";
 }
 .e-italic-icon:before {
 content: "\e710"
 }
}
```

```

.e-bullets-icon:before {
 content: "\e711";
}
.e-numbering-icon:before {
 content: "\e70e";
}
.e-ascending-icon:before {
 content: "\e70f";
}
.e-descending-icon:before {
 content: "\e707";
}
</style>
<script type="text/javascript">
 var toolbarObj = new ej.navigations.Toolbar({ width: 300, overflowMode:
 "Popup" });
 toolbarObj.appendTo('#template_toolbar');
</script>

```

### **BUTTONTEXT.CS**

```

public ActionResult Index()
{
 return View();
}

```

### **Integrate menu component**

You can integrate menu component as toolbar item in Toolbar using content template property. Menu can be populated with items as needed.

### **CSHTML**

```

@{
 List<object> menuItems = new List<object>();
 menuItems.Add(new
 {
 text = "Appliances",
 items = new List<object>()
 {
 new {
 text= "Kitchen",
 items = new List<object>()
 {
 new { text= "Electric Cookers" },
 new { text= "Coffee Makers" },
 new { text= "Blenders" },
 new { text= "Microwave Ovens" }
 }
 },
 new {
 text= "Television",
 items = new List<object>()
 {
 new { text= "Our Exclusive TVs" },
 new { text= "Smart TVs" },

```

```

 new { text= "Big Screen TVs" }
 },
 new { text= "Washing Machine" },
 new {
 text= "Refrigerators",
 items = new List<object>()
 {
 new { text= "Inverter ACs" },
 new { text= "Split ACs" },
 new { text= "Window ACs" }
 }
 },
 new { text= "Air Conditioners" },
 new { text= "Water Purifiers" },
 new { text= "Air Purifiers" },
 new { text= "Chimneys" },
 new { text= "Inverters" },
 new { text= "Healthy Living" },
 new { text= "Vacuum Cleaners" },
 new { text= "Room Heaters" },
 new { text= "New Launches" }
 }
});
menuItems.Add(new
{
 text = "Accessories",
 items = new List<object>()
 {
 new {
 text= "Mobile",
 items = new List<object>()
 {
 new { text= "Headphones" },
 new { text= "Batteries" },
 new { text= "Memory Cards" },
 new { text= "Power Banks" },
 new { text= "Mobile Cases" },
 new { text= "Screen Protectors" },
 new { text= "Data Cables" },
 new { text= "Chargers" },
 new { text= "Selfie Sticks" },
 new { text= "OTG Cable" }
 }
 },
 new { text= "Laptops" },
 new {
 text= "Desktop PC",
 items = new List<object>()
 {
 new { text= "Pendrives" },
 new { text= "External Hard Disks" },
 new { text= "Monitors" },
 new { text= "Keyboards" }
 }
 },
 new {

```

```

 text= "Camera",
 items = new List<object>()
 {
 new { text= "Lens" },
 new { text= "Tripods" }
 }
 }
}

});
menuItems.Add(new
{
 text = "Fashion",
 items = new List<object>()
 {
 new { text = "Men" },
 new { text = "Women" }
 }
});
menuItems.Add(new
{
 text = "Home & Living",
 items = new List<object>()
 {
 new { text= "Furniture" },
 new { text= "Decor" },
 new { text= "Smart Home Automation" },
 new { text= "Dining & Serving" }
 }
});
menuItems.Add(new
{
 text = "Entertainment",
 items = new List<object>()
 {
 new { text= "Televisions" },
 new { text= "Home Theatres" },
 new { text= "Gaming Laptops" }
 }
});
menuItems.Add(new
{
 text = "Contact Us",
});
menuItems.Add(new
{
 text = "Help",
});
}
@ (Html.EJS().ToolBar("defaultToolbar")
 .ContentTemplate(
 @<div>
 <div><button class='e-btn e-tbar-btn'>Cut</button> </div>
 <div><button class='e-btn e-tbar-btn'>Copy</button> </div>
 @Html.EJS().Menu("menu").Items(menuItems).Render()
 <div><button class='e-btn e-tbar-btn'>Paste</button> </div>
 </div>
)
)

```

```
.Render()
)
```

### **MENUCOMPONENT.CS**

```
public ActionResult Index()
{
 return View();
}
```

### Accessibility in ASP.NET MVC Toolbar control

The Toolbar control has been designed, keeping in mind the [WAI-ARIA](#) specifications, and applying the WAI-ARIA roles, states, and properties along with keyboard support for people who use assistive devices. WAI-ARIA accessibility support is achieved through attributes like `aria-label`, and `aria-orientation`. It provides information about elements in a document for assistive technology. The control implements keyboard navigation support by following the [WAI-ARIA practices](#), and has been tested in major screen readers.

The accessibility compliance for the Toolbar control is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

```
margin: 0.5em 0;
```

```
}
```

```
</style>
```

```
<div> - All features of the control meet the requirement.</div>
```

```
<div> - Some features of the control do not meet the requirement.</div>
```

```
<div> - The control does not meet the requirement.</div>
```

### ARIA attributes

ToolBar control is designed by considering [WAI-ARIA](#) standard. Toolbar is supported with ARIA Accessibility which is accessible by on-screen readers, and other assistive technology devices. The following list of attributes are added in the Toolbar.

#### | Property | Functionalities |

```
| --- | --- |
```

```
| role="toolbar" | Attribute is set to the ToolBar element describes the actual role of the element. |
```

```
| aria-orientation | Attribute is set to the ToolBar element to indicates the ToolBar orientation. Default value is horizontal. |
```

```
| aria-label | Attribute is set to ToolBar element describes the purpose of the set of toolbar. |
```

```
| aria-expanded | Attribute is set to the ToolBar Popup element to indicates the expanded state of the popup. |
```

```
| aria-haspopup | Attribute is set to the popup element to indicates the popup mode of the Toolbar. Default value is false. When popup mode is enabled, attribute value has to be changed to true. |
```

```
| aria-disabled | Attribute set to the ToolBar element to indicates the disabled state of the Toolbar. |
```

### Keyboard interaction

Keyboard navigation is enabled by default. Possible keys are

```
| Key | Description |
```

```
|-----|-----|
```

```
| Left | Focuses the previous element. |
```

```
| Right | Focuses the next element. |
```

```
| Enter | When focused on a Toolbar command, clicking the key triggers the click of Toolbar element. When popup drop-down icon is focused, the popup opens. |
```

```
| Esc(Escape) | Closes popup. |
```

```
| Down | Focuses the next popup element. |
```

```
| Up | Focuses the previous popup element. |
```

```
| Home | Moves focus to the first Toolbar. |
```

| End | Moves focus to the last Toolbar. |

| Tab | To Move focus through the interactive elements. |

| Shift + Tab | To Move focus through the interactive elements. |

#### Ensuring accessibility

The Toolbar control accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Toolbar control is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Toolbar control with accessibility tools.

See also

- [Accessibility in Syncfusion components](#)

#### CSS Structure in ASP.NET MVC Toolbar Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on user preference.

##### Customizing Toolbar

Use the following CSS to customize the Toolbar.

```
`CSS
.e-toolbar {
border: 5px solid rgb(173, 255, 47);
}
`
```

##### Customizing the Toolbar items

Use the following CSS to customize the items of Toolbar.

```
`CSS
.e-toolbar .e-toolbar-item {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```

Use the following CSS to customize the button in the items of Toolbar.

```
`CSS
.e-toolbar .e-tbar-btn {
background: #add8e6;
border: 1px solid #5a70cc;
}
`
```



,

### Customizing Toolbar's item icon

Use the following CSS to customize the item icon of Toolbar control.

`CSS

```
.e-toolbar .e-tbar-btn .e-icons {
background: #185655;
color: #d7f9d4;
}
```

,

### Customizing the hover state of Toolbar control

Use the following CSS to customize the toolbar item when hovering.

`CSS

```
.e-toolbar .e-tbar-btn:hover {
background: #c0e3a1;
border: 1px solid green;
}
```

,

### Customizing selected item of Toolbar control

Use the following CSS to customize the selected toolbar item.

`CSS

```
.e-toolbar .e-tbar-btn:focus {
background: #add8e6;
border: 1px solid #5a70cc;
}
```

,

## How To

### Set command customization

The `htmlAttributes` property of the Toolbar item is used to set the HTML attributes ('ID', 'class', 'style', 'role') for the commands.

When style attributes are added, if the same attributes were already present, they will be replaced. This is not so in the case of `class` attribute. Classes will be added to the element instead of replacing the existing ones.

Single or multiple CSS classes can be added to the Toolbar commands using the Toolbar item `cssClass` property.

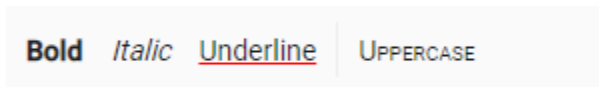
### CSHTML

```
@Html.EJS().ToolBar("defaultToolbar").Items(ViewBag.Items).Render()
```

### CUSTOMIZATION.CS

```
public ActionResult Index()
{
 List<ToolBarItem> items = new List<ToolBarItem>();
 items.Add(new ToolBarItem { Text = "Bold", Type = ItemType.Button ,
 HtmlAttributes = new HtmlAttributes { Class = "custom_bold", Id = "itemId" }
 });
 items.Add(new ToolBarItem { Text = "Italic", HtmlAttributes = new
 HtmlAttributes { Class = "custom_italic" } });
 items.Add(new ToolBarItem { Text = "Underline", HtmlAttributes = new
 HtmlAttributes { Class = "custom_underline" } });
 items.Add(new ToolBarItem { Type = ItemType.Separator });
 items.Add(new ToolBarItem { Text = "Uppercase", CssClass = "e-txt-
 casing" });
 ViewBag.Items = items;
 return View();
}
public class HtmlAttributes
{
 public string Class { get; set; }
 public string Id { get; set; }
}
```

Output be like the below.



Set Essential JS 2 Tooltip to the commands

The `tooltipText` property of the Toolbar item is used to set the HTML Tooltip to the commands that can be viewed as hint texts on mouse hover.

Initialize the Tooltip with the Toolbar target. Refer to the following code example:

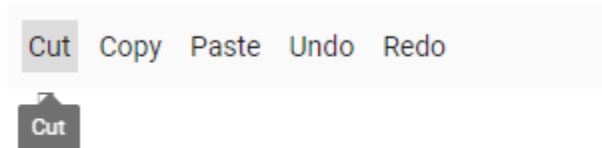
### CSHTML

```
@(Html.EJS().ToolBar("defaultToolbar")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Type = ItemType.Button, TooltipText = "Cut", Text
 = "Cut" },
 new ToolBarItem { Type = ItemType.Button, TooltipText = "Copy", Text
 = "Copy" },
 new ToolBarItem { Type = ItemType.Button, TooltipText = "Paste",
 Text = "Paste" },
 new ToolBarItem { Type = ItemType.Button, TooltipText = "Undo", Text
 = "Undo" },
 new ToolBarItem { Type = ItemType.Button, TooltipText = "Redo", Text
 = "Redo" }
 })
 .Render()
)
```

**TOOLTIP.CS**

```
public ActionResult Index()
{
 return View();
}
```

Output be like the below.

**Set item-wise custom template**

The Toolbar supports adding template commands using the `template` property. Template property can be given as the `HTML element` that is either a `string` or a `query selector`.

**As a string**

The HTML element tag can be given as a string for the template property. Here, the checkbox is rendered as a HTML template.

`typescript

template: "<div><input type='checkbox' id='check1' checked=''>Accept</input></div>"

,

**As a selector**

The template property also allows getting template content through query `selector`. Here, checkbox 'ID' attribute is specified in the template.

`typescript

template: "#Template"

,

**CSHTML**

```
@using Syncfusion.EJ2.Navigations;
<div>
 <div>
 @ (Html.EJS().ToolBar("defaultToolbar")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Text = "Cut" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Text = "Paste" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Template = "<div><input type='checkbox'
id='check1' checked=''>Accept</input></div>" },
 new ToolBarItem { Text = "Undo" },
 new ToolBarItem { Text = "Redo" },
 new ToolBarItem { Template = "#Template" }
 })
)
 </div>
</div>
```

```

 })
 .Width("330")
 .Render()
)
</div>
<button id='Template' class='e-btn'>Template</button>
</div>

```

### SELECTOR.CS

```

public ActionResult Index()
{
 return View();
}

```

### Add Toggle Button in ToolBar Control

ToolBar supports to add a toggle Button by using the template property. Refer below steps

- By using Toolbar template property, pass required HTML String to render toggle button.

`typescript

```
template='<button class="e-btn" id="media_btn"></button>'
```

,

- Now render the toggle Button into the targeted element in Toolbar created event handler and bind click event for it. On clicking the toggle Button, change the required icon and content based on current active state.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
@(Html.EJS().ToolBar("defaultToolbar")
 .Created("created")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Template = "<button class='e-btn'
id='media_btn'></button>" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Template = "<button class='e-btn'
id='zoom_btn'></button>" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Template = "<button class='e-btn'
id='undo_btn'></button>" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Template = "<button class='e-btn'
id='filter_btn'></button>" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Template = "<button class='e-btn'
id='visible_btn'></button>" },
 })
 .Width("330px")
 .Render()
)

```

```

<style>
 /* Toolbar Styles */
 .e-hide-icon::before {
 content: '\eb23';
 }
 .e-filter-icon::before {
 content: '\e946';
 }
 .e-undo-icon::before {
 content: '\ebc5';
 }
 .e-play-icon::before {
 content: '\e328';
 }
 .e-zoomin-icon::before {
 content: '\ec31';
 }
</style>
<script type="text/javascript">
 function created() {
 zoomBtn = new ej.buttons.Button({ cssClass: `e-flat`, iconCss: 'e-
icons e-zoomin-icon', isToggle: true });
 zoomBtn.appendTo('#zoom_btn');
 mediaBtn = new ej.buttons.Button({ cssClass: `e-flat`, iconCss: 'e-
icons e-play-icon', isToggle: true });
 mediaBtn.appendTo('#media_btn');
 undoBtn = new ej.buttons.Button({ cssClass: `e-flat`, iconCss: 'e-
icons e-undo-icon', isToggle: true });
 undoBtn.appendTo('#undo_btn');
 filterBtn = new ej.buttons.Button({ cssClass: `e-flat`, iconCss: 'e-
icons e-filter-icon', isToggle: true });
 filterBtn.appendTo('#filter_btn');
 visibleBtn = new ej.buttons.Button({ cssClass: `e-flat`, iconCss:
'e-icons e-hide-icon', isToggle: true, content: 'Hide' });
 visibleBtn.appendTo('#visible_btn');
 document.getElementById('zoom_btn').onclick = () => {
 if (document.getElementById('zoom_btn').classList.contains('e-
active')) {
 zoomBtn.iconCss = 'e-icons e-zoomout-icon';
 } else {
 zoomBtn.iconCss = 'e-icons e-zoomin-icon';
 }
 };
 document.getElementById('media_btn').onclick = () => {
 if (document.getElementById('media_btn').classList.contains('e-
active')) {
 mediaBtn.iconCss = 'e-icons e-pause-icon';
 } else {
 mediaBtn.iconCss = 'e-icons e-play-icon';
 }
 };
 document.getElementById('undo_btn').onclick = () => {
 if (document.getElementById('undo_btn').classList.contains('e-
active')) {
 undoBtn.iconCss = 'e-icons e-redo-icon';
 } else {
 undoBtn.iconCss = 'e-icons e-undo-icon';
 }
 };
 }
}

```

```

 }
 };
 document.getElementById('filter_btn').onclick = () => {
 if (document.getElementById('filter_btn').classList.contains('e-active')) {
 filterBtn.iconCss = 'e-icons e-filternone-icon';
 } else {
 filterBtn.iconCss = 'e-icons e-filter-icon';
 }
 };
 document.getElementById('visible_btn').onclick = () => {
 if
 (document.getElementById('visible_btn').classList.contains('e-active')) {
 document.getElementById('content').style.display = 'none';
 visibleBtn.content = 'Show';
 visibleBtn.iconCss = 'e-icons e-show-icon';
 } else {
 document.getElementById('content').style.display = 'block';
 visibleBtn.content = 'Hide';
 visibleBtn.iconCss = 'e-icons e-hide-icon';
 }
 };
}
</script>

```

### TOGGLEBUTTON.CS

```

public ActionResult Index()
{
 return View();
}

```

### How to customize toolbar scrollStep

ToolBar supports to customize the scrolling distance when you click the left and right side navigation icons. we can customize **ScrollStep** property for scrolling distance. Refer to the following code example.

By using Toolbar scrollStep property, pass a required value to customize toolbar scrollStep.

### CSHTML

```

@using Syncfusion.EJ2.Navigations;
@(Html.EJS().ToolBar("defaultToolbar")
 .Items(new List<ToolBarItem> {
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-cut-icon tb-icons", TooltipText = "Cut" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-copy-icon tb-icons", TooltipText = "Copy" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-paste-icon tb-icons", TooltipText = "Paste" },
 new ToolBarItem { Type = ItemType.Separator },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-bold-icon tb-icons", TooltipText = "Bold" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-underline-icon tb-icons", TooltipText = "Underline" },
 new ToolBarItem { Type = ItemType.Button, PrefixIcon = "e-italic-icon tb-icons", TooltipText = "Italic" },
 })
)

```

```

 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-color-icon
tb-icons", TooltipText = "Color-Picker" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-alignleft-
icon tb-icons", TooltipText = "Align-Left" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-
alignright-icon tb-icons", TooltipText = "Align-Right" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-
aligncenter-icon tb-icons", TooltipText = "Align-Center" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-
alignjustify-icon tb-icons", TooltipText = "Align-Justify" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-bullets-
icon tb-icons", TooltipText = "Bullets" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-numbering-
icon tb-icons", TooltipText = "Numbering" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-ascending-
icon tb-icons", TooltipText = "Sort A - Z" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-
descending-icon tb-icons", TooltipText = "Sort Z - A" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-upload-
icon tb-icons", TooltipText = "Upload" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-download-
icon tb-icons", TooltipText = "Download" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-indent-
icon tb-icons", TooltipText = "Text Indent" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-outdent-
icon tb-icons", TooltipText = "Text Outdent" },
 new ToolbarItem { Type = ItemType.Separator },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-clear-icon
tb-icons", TooltipText = "Clear" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-reload-
icon tb-icons", TooltipText = "Reload" },
 new ToolbarItem { Type = ItemType.Button, PrefixIcon = "e-export-
icon tb-icons", TooltipText = "Export" },
 })
 .Width("600")
 .ScrollStep("50")
 .Render()
)
<style>
 @@font-face {
 font-family: 'Material_toolbar';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAKAIAAAwAgTlMvMjltShMAAAEoAAAVmNtYXDoMOjqAAACDAAAHHnbHlmIuy
19QAAAswAACNMAGVhZA6okZMAAADQAAAAANmhoZWEIUQQkAAAArAAAACRobXR4jAAAAAAAYAAAC
MbG9jYYc0kUIAAAKEAAAAASG1heHABOwG8AAABCAAAACBuYW11x/RZbQAAJhgAAAKRcG9zdJZeEVU
AACisAAACGAABAAAEAAAAAFwEAAAAAAD9AABAAAAAAAAAAAAAAAAAAAAAAIwABAAAAAQAAQsu/F8
PPPUACwQAAAAAANXLJlEAAAAA1csmUQAAAAAD9AP0AAAACAACAAAAAAAAAAAAEAAAAjAbAADgAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQQA ZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnIQQAAAAAXAQAAAAAAAAABAAAAAABAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAA
AAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAA
AAAQAAAAEAAAABAAAAAQAAAAAACA AAAAwAAABQAawABAAAFAAEAGQAAAAEAAQAA5yH//wA

```

A5wD//wAAAAEABAAAAEAAgADAAQABQAGAAcACAAJAAoACwAMAA0ADgAPABAAEQASABMAFAAVABY  
AFwAYABkAGgAbABwAHQAEAB8AIAAhACIAAAAAADIAjgFwAfgCIAKYAxIDSAO2BRYFMAVcBnIGugb  
2ByoHQgguCNYJRgn6CiQKiAQuCsgMFgzADOYNzg7WDvAQYBEyEaYABwAAAAAD9APzAAMABwAKAA4  
AEgAVABkAADchNSElITUhJTkBBSelITUhNSEFFxEnITUhDAPo/BgBtgIy/c7+SgG2AjL9zgIy/c7  
+Svr6A+j8GAXefV67Pl19Xvr6AfScXgAAAAIAAAAA/QD9AAEAEGAACUhNxc3AREfDyE/DxEvDyE  
PDgOF/PbDisP9gQEBawQEBgYICAgJCgoLCwsDCgsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAGICQo  
KCwsL/PYLcwsKCgkICAgGBgQEAWGz+qf6AYX89gsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAGICQo  
KCwsLAwoLCwsKCgkICAgGBgQEAWEBABQEDBAQGBgGICAKKCgsLAAACAAAAAPzA/QAQAC/AAABFQ8  
PLw8/Dx8OAQ8ELwErAQ8FFR8FBxcPAXUFbzBNx8LowI/Cx8BOWE/Bj0BLwQ/Aic/BC8HKwEHLws  
rAg8FARIBAgUGBwkDAwODxAQERITEhIREQ8PDgWMCgkHBgUCAQECBQYHCQoMDA4PDxEREhITEhE  
QEAS8ODAwKCQCGBQL+zxUWFhUWfWUfBAUDBANqAgEBAGIDbgMDBwMCAQEBAAMkDBAQEBQSEFBYWFxQ  
CAGIDBAQEBCwFBAQEAWICAhQXFRYVgAQFBQQAeAwNoAgEBAGIDcAEBAQNvAgIBAQEBA2gDBAQFBW  
DFBYWFxIBAgMDAwQFBcWFBABQDBAICAgJCRIQEBAODgWLCgkHBgQDAQEDBAYHCQoLDA4OEBAQEhI  
SEhAQEA4ODAsKCQCGBAMBAQMEBgcJCgsMDg4QEBASAc6ECwwNDjIBAQICA7QEBQQFBAMEUjIyVgM  
EBAQFBWAwAwICATMODQwLhAQEBAMCAGECAGIDBAMEhASMDQ4yAQECAGOWBAQFBUEAwRSDAwAMLY  
DBAQEBQQFsAMCAGeZDg0MC4QEAWQDAgICAgICAwQDAAAAAMAAAAA/MD2AAyADUAaQAAJRUFDTs  
BPw41LwgPBwMhAScXAQ8GHQEfBQEfBjSBPWyBPWyvBwEDFgIDBAQGBgcICQkKCgsLCwsLCwoKCQg  
ICAYGBAQDAQEDBACMCQoLFCMtFQoJCQCfBHv96gEL04X+4gYFBAQCAgICAgIEBAUBNwCHBwgHCAC  
IBwgHCAGHBwYBOAUEAwMCAQEBAQIDAQFbV4PlwsLCwoKCQkIBwYGBAQDAgIDBAQGBgcICQkKCgs  
LCwcPEBAYEBAPHck3HRAQEBAQEAEIAQRThf7iBgCIBwgHCAGICAgHBwCH/skGBQQEAWIBAQIDBAQ  
FBgE3BwCHBwgICAgHCAGHCAGAfEABQAAAAAD9APzAAMABwALAA8AEwAANYe1ITChNSEnITUhNyE  
1ISchNSEMA+j8GN4CLP3U3gPo/BjeAiz91N4D6PwYDF6AW5xefVqAXgAAAAEAAAAAAP0A/QACQA  
TABCAWwAAQcVMzcxMzUjLwEjFTMbATM1IwE1ESERBxEfDyE/DxEvDyEPDgFro8ObnnROxOp0nZv  
qTij+8AGW/NReAQEDBAQGBgGICAKKCgsLCwMKCwsLCgoJCAGIBgYEBAMBAQEBAWQEBgYICAgJCgo  
LCwv89gsLCwoKCQgICAYGBAQDAQENAYOWli04BSUBK/7VJQFSffzUAYwR/PYLcwsKCgkICAgGBgQ  
EAWEBABQEDBAQGBgGICAKKCgsLCwMKCwsLCgoJCAGIBgYEBAMBAQEBAWQEBgYICAgJCgoLCwAAAA  
CAAAAAAOWA/QAAwBpAAA3ITUhExUfHTsBPx01ESMRDw8vDxEjagMs/NRKAgIDAwUFBgcHCAkJCgs  
LCwvNDQ0ODw4PEAS8QERAREREREBEQDxAPDg8ODQ0NDASLCwoJCQgHBwYFBQMDAgKLAQMFbGgKCw  
ODxARERMTFBQTEEXEREAS8ODAsFCQCGBAKLDH0BSBEREREQEAS8QDg8ODg0MDQsLCwoJCQgHBwYFBQ  
DAgEBAGMEBQUGBgGICQkKCgsMDAwNDg4ODw8PEBAQERERAB7+RRQTEhIREAS8NDQsKCAyFAwEBAW  
GCAOLDQ0PCBASEhMTACUABQAAAAAD9APXAAIABQANABcAGgAAJTCjASM3ATM3MxczAyMFIQEVIITU  
hATUhJTMnAgJx4wG/vl/+Fot+ila3FD9RgEg/t4Bov7UAST+aAF/6XQobQET//47eHgCM07+XD5  
NAaQ/UHMAAAAAQAAAAAD9ALoAF8AABMhJz8PHx03Lx8PDycMABWyDQ0ODg8PDxAQEBERERIREhA  
QEBAQDw8PDw4ODg0NDQwMFxYTEhAHBgYGBXUHBwgJCQoLCwvNDQ0PDg8QEBERERITEhMUEXQVFB  
VFRgYFxcXfHYVfHQUFBMTEhGwARi6CwsJCgGICAYGBgQEAWIBAQEBAgIDBAQFBQYHBwgICQkKFRY  
YGHsODg8PDygUFBMTEhISERARDw8PDg0NDASLCgoICAgGBgQEAWMBAQECAwQFBgcJCQoLDA0ODg+  
6AAYAAAAA/MD9AA/AGsAqWDrAO8BMwAAARUFDTsBPw09AS8ODw41Hwk7AT8IPQEvByMnByMPByU  
fDz8PLw8PDiUfDz8OPQEvDSsBDw01ESERBxEfDyE/DxEvDyEPDgHhAgMFBQYHCAkKCgsLDA0NDA0  
MCwsKCgkIBwYFBQMCAGMFBQYHCAkKCgsLDA0MDQ0MCwsKCgkIBwYFBQMC/scBAQEfbWgKCwYGBwY  
GBgWKCACFAQEBAQUHCAoMBgYGBwYGCwoIBwUBAQHzAQECBAQEByYGCACICQkJCgoJCQgJBwgGBgY  
EBAMDAQEBAQMDBAQGBgYIBwkICQkKCgkJCQgHCAYGBgQEBAIB/qgBAQMEBAYGBwgICQoKCgsLCws  
LCgkJCQCHBwUFAwMCAgMDBQUHBwcJCQkKCwsLCwsKCgoJCAGHBgYEBAMBA1D81F4BAQMDBAUGBgC  
HCAkJCQkKAYYKQKJCQgHBwYGBQQEAgEBAQECAwQFBgYHBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwM  
BAWQNDAwMCwoKCQgHBgUFAwICAwUFBgcICQoKCwvMDA0NDALCwsJCQgHBwUEAWIBAQIDBAUHBwg  
JCQsLCwvMMQYGBgsKCQCEAgEBAGQHCQoLBgYGBwYGCwoJBgUCAQECBQYJCgsGBvMJCgkICAgHBwY  
FBQDAWEBABQEDAwQFBQYHBwgICAKKCQoJCQkICACGBwUFBAMCAQEBAQIDBAUFbWYHCAGJCQkGCws  
KCgoJCAGHBgYEBAMBAQEBAWQEBgYHCAGJCgoKCwsLCwsKCQkJBwCHBQUdAwICAwMFBQCHBwkJCQo  
LC9/81AMsA/zaCgkJCQkIBwcGBgUEBAIBAQEDAgQEBOUHBwcICQkJCQoDJgoJCQkJCAChBwUFBABQ  
CAGCEIEBAUFbWCHCAkJCQkAAAAAAGAAAAADTQP0AAMACgAANYe1IQEjCQEjESFKA2z81AEG7AG  
cAZzs/qAMfQIK/mQBnAFhAAYAAAAA/QD8wADAACACwAPABIAfGAANYe1ISuHNSelITUhNSE1KQE  
RNwMhNSEMA+j8GAG2AjL9zgIy/c4CMv30/kr6+gPo/BgMXn1efV19Xv4M+gGWXgAFAAAAAAPzA/M  
AJQBpAKgArADwAAABFT8bIw8GBR8PNSMvDT8CJw8OHw8TByMPDBc/AzsBHwUzHwYzLxcjJREhEQc  
RHw8hPw8RLw8hDw4CKg8QDw8ODg4NDQwMDAwKCwoKCQgJDw0KCQQDAgLxBAUGBgGJC/7WDQ0NDg4  
PDw8PEAS8QEBAQEAOKCQkJCQgIBgQEBAUDAQEDBKsKCQgIBwYGBQUdAwMBAQEBAQEDAwQEBQYHBwg  
ICgoL9g4OHA4ODQ4NDg0NDQwNDKkLDA8HCAkJCAGICA4DEAUFBQMEAU4EBwkKDQ8REwoKCwsMDAw  
NDQ4ODg4PDy8KAZP81F4BAQMDBAUGBgCHCAkJCQkKAYYKQKJCQgHBwYGBQQEAgEBAQECAwQFBgY  
HBwgJCQkJCvzaCgkJCQkIBwcGBgUEAwMBAZz0AgMDBAQFBQYHBwgICQkJCgsLCwsYGHsbDw4PDwo



KCQgIBwaUDAsLCgkIBwcGBQQEAgIBafEBAwMEBQYIBQcGBw8PEA8ODa0NDQ00Dg4ODw8PDw8PEA8  
PEA8PEA8ODw8ODg00DQ0MAj8BBAMDBAQFBgChBwkJCqoGBQQBAQICBAMJEAcHCAGJCh0dHRsaGRc  
XCgoKCQgICAcGBgYEBQMDBj781AMsA/zaCgkJCQkIBwcGBgUEAwMBAQEBAgQEBQYGBwICQkJCQo  
DJgoJCQkJCAChBgYFBAQCAQEBAQIEBAUFBwCHCAkJCQkAAgAAAAAD8wPrAB8AMwAAEw8HHww/BBU  
hNSEBNwkBPwcvCDYKCAcGBQMCAQECAwUGBwgKrwgJCgoKCgkINQKQ/Xv+20EBPAGOCgkHBgUDAgE  
BAgMFBGcJCtMBoQsMDQ0NDg4ODg00DQ0NDaUvBgUDAQECCBAY0I14BJEH+XAGRCwwMDQ00Dg4ODg4  
NDQwMC9QABgAAAAAD9AP0AAMADwATAB0AIQAnAAALITUhIzMVIXUzFSMVMzUjNyE1ISMzBxUzNSM  
3NSM3ITUhJzVMVzUjAQYC7v0S+n0/P328vPoC7v0S+nh4vHh4vPoC7v0S+j4/fWpeID4gPvrbXXw  
/P3w/u14gvPoAAAAABQAAAAAD9APbAAIABQANABcAGgAAJTCjAyM3ATM3MxczAyMFIQEIVITUhATU  
hJzMnAgJx4x6+Xv76Wi39LF3fTwFLAST+3AGk/tIBJP5mw+10JXMBGP/+N3h4AjRQ/lo+TQGpPk5  
zAAABAAAAA0sA/QACwAAATMDIXUhNSMTMzUhAXGd88gCPJ3zyP3EAx79xNbWajzWAAAGAAAAAAP  
0A9QAawBDAECahwCLAMsAACUhNSEHFR8OPw49AS8ODw4TITUhBxUfDTsBPw09AS8ODw4TITUhBxU  
fDj8OPQEvDg8OAQYC7v0S+gIBAwMEBQUFBGcGCACICAgICAcHBgYGBQQEAWMCAQECAwMEBAUGBgY  
HBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBGcGCACICAgICAcHBgYGBQQEAWMCAQE  
CAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAvoC7v0S+gIBAwMEBQUFBGcGCACICAgICAcHBgY  
GBQQEAWMCAQECAwMEBAUGBgYHBwgICAgIBwgGBwYFBQUEAwMBAkpeLwgIBwCHBwYFBQUDBAICAQE  
BAQICBAMFBQUGBwCHBwgICAgIBwcGBgYFBAQDAWIBAQEBAgMDBAQFBgYGBwICAFgXS4ICAgHBwY  
GBgUEBAMDAgEBAgMDBAQFBgYGBwICAgICAcHBwCHBQUFAwQCAgEBAQECAgQDBQUFBGcHBwCIAUB  
dLggICAcHBgYGBQQEAWMCAQEBAQIDAQEBQYGBGcHCAgICAgHBwCHBQUFBQMEAgIBAQEBAgIEAwU  
FBQYHBwCHCAAAwAAAAADmQP0AAcAKACNAAABFSE1MxEhESUHFQ8GLwc/Bx8GJysBDw0VERUfDTM  
hMz8NNRE1Lw0rAS8OKwEPDQEdAcZb/YQBbAEDBAYHBwkJCQkHBwYEAwEBAwQGBwJCQkJBwCGBAO  
svwkJCQgICAcGBgYEBAMCAgICAwQEBgYGBwgICAKJCQJ8CQkJCAgIBwYGBgQEAWICAgIDBAQGBgY  
HCAgICQkJvwmFBQYGBwgICQkJCgoKCwsLCwoKCgkJCQgIBwYGBQUdPoiI/SkC1y4FBQgIBwUEAwE  
BAwQFBwgICgkICAcFBQIBAQIFBQcICCCQCAgMEBAYGBGcICAgJCQn9KQkJCQgICAcGBgUFBAMCAgI  
CAwQFBQYGBwgICAKJCQLXCQkJCAGIBwYGBgQEAWICCGkJCAGIBwYGBQQEAWICAgIDBAQFBgYHCAg  
ICQkAAQAAAAAD9ALoAGAAAAExLw8PHxc/Gh8PBYERA0QREhMTFBQUFRYWFhcxgYFRUVFBUUEXQ  
TEhMSEREREBApDg8NDQ0MCwsKCQkIBwd1BQYGBGcQEhMWFwMDQ0NDg4ODw8PDxAQEBAQEhESERE  
QERAQDw8PDg4NDbIBtQIUdW4ODQwLCgkJBwYFBAMCAQEBAwMEBAYGCAGICgoLCwWMDQ4PDw8REBE  
SEhITExQUKA8PDw4OGxoYFhUKCQkICAcHBgYFBAQDAgIBAQEBAgMEBAYGBGgICAOJCwu6AdAAAA  
AAAAAAAP0A/MAAgAFAAgACwAQABQAFwAbAB4AIQAPAC0AMQB1AAABETclFzUXNyMFNyETfQUhEQE  
hJRM1MycFMSEnBzcnBxcRBRMDBSUDEY0BEQmLIWUDEQcRHw8hPw8RLw8hDw4CGcj+ZaG3MJb+wM7  
+4jQBCv6EAy7+ggEKdP1S3JkBCwEjWemWlvrIATJ0dP7n/up3dwEWAzhy/vQ0/vZyXgEBAwQEBgY  
ICAgJCgoLCwsDCGsLCwoKCQgICAYGBAQDAQEBAQMEBAYGCAGICQoKCwsL/PYLCwsKCgkICAgGBgQ  
EAwEBxv7fWSQ730Bky8v+9QNxAyH+f28BHx2ZmcukmTgJyWEEp/7n/ud3dweZAR13Bv5xAR1ycv7  
yAYAR/PYLCwsKCgkICAcHBQUEAwEBAQEDBAUFBwICAKKCgsLCwMKCwsLCgoJCAgHBwUFBAMBAQE  
BAwQEBgYIBwkJCgoLCwAAAAFAAAAAAP0A/MAAwAHAASADwATAAA3ITUhJSE1ISUhNSE1ITUhJSE  
1IQwD6PwYAVgCkP1w/qgD6PwYAVgCkP1w/qgD6PwYDF6AW5xefV19XgAAAAAKAAAAAAP0A/MAAwA  
HAASADwATABcAGwAfACMARwAAARUjNSMVIzUjFSM1ARUjNSMVIzUjFSM1JRUjNSMVIzUjFSM1JxE  
fByE/BxEvByEPBgOW+j7bP9oDLPo+2z/aAyz6Pts/214BAwUGAwgJCgOJCgkJBwYDBAIBAwUGAwg  
JCvx3CgkJBwYFAwElvb27u7u7ARrb29vb29v6vLy8vLy8hvyCCwoJBwQGBAIBAwUHBwUJCgOECwo  
JBwQGBAIBAwUGCAKAAAAAUAAAAA/QD8wADAACACwAPABMAADchNSE1ITUhNSE1ITUhNSE1ITU  
hDAPo/BgCkP1wA+j8GAKQ/XAD6PwYDF6BV59efVqAXgAAAAADAAAAAAP0A00AAwAHAASAADchNSE  
1ITUhNSE1IQwD6PwYA+j8GAPo/Bizb6Zwpm8AAAAABQAAAAAD9AP0AD8AXwcFAKQBIgAAJQ8PLw8  
/Dx8OExUPBSsBLwU9AT8FOWefBQMPDy8PPw8fEAE1IWUVHw8zPwMXBy8FDw8fDz8PNS8DNwEzNQE  
/BS8PDw4BOAEBAwMEBQYGBwgICQkKCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgo  
KCgkJCAGHBgYFBAMDAeICAgMDBQUFBQUFAwMCAgICAwMFBQUFBQUdAwIC4QEBAwMEBQYGBwgICQk  
KCgoKCgoJCQgIBwYGBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAGHBgYFBAMDAftkAV6W/K4  
BAwUHCaOMDQ4PERETEXQUCwsVFBn2dgkKCgoVFhQUExMREQ8ODQwKCAcFAwEBAwUHCAOMDQ4PERE  
TEXQUFBQTEEXERDw4NDAoIBwUDAQEEBgD2AV6W/ZYFBAMCAwEBAwUHCAOMDQ4PERETEXQUFBQTEEXE  
RDw4NDAoIBwUD1AoKCgkJCAGHBgYFBAMDAQEBAQMDBAUGBgICAKJCgoKCgoKCQkICAcGBgUEAwM  
BAQEBAwMEBQYGBwgICQkKCgEiBQUFAwMCAgICAwMFBQUFBQUdAwICAgIDAwUFAScKCgoJCQgIBwY  
GBQQDAwEBAQEDAwQFBgYHCAgJCQoKCgoKCgkJCAGHBgYFBAMDAQEBAQMDBAUGBgICAKJCgqgZAF  
eMpYKChQTEEXERDw4NDAoIBwUDAQEEBgD2dgUEAwIDAQEDBQcICgWNDg8RERMTFBQUFBMTEREpDg0  
MCgghBQMBAQMFBwgKDA00DxERExMUFASLFRQTdv6iMgJqCQoKChUWFBQTEEXERDw4NDAoIBwUDAQE  
DBQcICgWNDg8RERMTFAADAAAAANXA7UAIGBFAJMAAAEzHw4PDIsBNRMzHw4PDIsBNQMhPxEvDz8  
PLxghAkGKCgkJCAGHBwYGBAQEAgEBAQEDAwQFBgYHBwgJCAkKCeDACgoJCQgIBwCGBgQEBAIBAQE  
BAgQEBAwYGBwICAKJCgrAwAHDDQwMDBcWFRMSEQ8NDAoHBgQBAQIDBAYHBwkKCgsNDA4ODwsLCgo

KCAgIBgYFBQMDAQEBAQECAwQEBAUGDA8QEhQVfGwMDA0NDQ0N/nABogICAwQEBgYGBwgICQkKCQo  
KCQgJBwgGBgUFBAMCArsBdwICAwQEBgYGCACICQkKCQoKCQkIBwgGBgYEBAMCArV9MQEBAQIGCAo  
MDg8RehQUFhcYGBERERAQE4ODgWMDAoJCQcICQkKCgoLDAsMDAwMDQwNDQwNDQwMCwwLCxQUERA  
ODQoFAwQDAgEBAQAABQAAAAAD9APzAAMABwALAA8AEwAANYe1ITUhNSE1ITUhNSE1ITUhNSEMA+j  
8GAPo/BgD6PwYA+j8GAPo/BgMXn1enF59XX1eAAAAAEEAAAAA9QD1ADUAAATHx8/DxcRIRcPDy8  
fPx8fDzMvHw8eKwECAwQFBggICQoMDA00DhAQERISExQUFRUWFhcXGBgYGBgXFxcWfHUVFBQTEhI  
REIr+ZrsMDA00Dg4PEBAQEBESERISEhIREhEQEQ8QDw8ODg0NDAwLCgoJCQgHBgYEBACQAQEBAQI  
EBAQGBgcICQkKCgsMDA0NDg4PDxAPERAREhESEhwcGxoAGBgWFRQSEQ8OCwp7BQYHCAGJCQoLCww  
NDQ4ODg8QEBERERISEhMTFBMUFRQYGBgXFxYWFUUFMBSEhEQEA4ODQwMCgkICAYFBAMCAgAYGBc  
XFxYWFUUFMBSEhEQEA4ODQ0LCgoICAYFBAMCAQECAwQFBggICgoLDQ00DhCKAZq7DAsLCgkJCAC  
HBQUEAwMBAQEBAgQEBAYGBwgICgkLCwwMDQ0ODg8PDxAREBESERISEhIREhEQERAPDw8ODg0NDAw  
LCwkKCAgHBgYEBACQAQECAwUICQsNDxASExUWFxgaExITERIREBAQDw8ODg0NDAsLCgoJCAcHBgY  
EBAMCAQEBAgMEBQYICAoKCw0NDg4QEBESEhMUFBUVFhYXFxcYAAAAAaGAAAAAD8gP0AGcA7gAAARU  
PGC8YPQE/FzsBHxcFHx8/DxcVATcBIyc/Dj0BLx0rAQ8dAoABAgIDAwQFBQUNDxATExYLCwwMDAw  
NDQ0NDQ0NDQwNDAsMCxUUEhAPDQUFBQDQAwMBAQEBAwMDBAUFBQ0PEBIUFQsMCwwNDA0NDQ0NDQ0  
NDAwMDAsLFhMTEA8NBQUFBAMDAgIB/Y0BAQMDBAUGBgICQkLCwsNDA4ODg8QEBARERISEhMTEExE  
REBEQEBAQDw8ODg4ODA0OAR1W/uMuDgoKCQkIBwYGBgQEAWMCAQICAwQFBgcHCAkKCgsMDA0NDg8  
PDxAREREREhMSExMTEExMSEhIRERAQE8ODg4MDQsLCwkJCAgGBgUEAwMBAoIODQ0MDQwMDAsLFRQ  
SEQ4NBgUEBAQDAgEBAQEBAQIDBAQEBAQYNDhESFBULCwwMDA0MDQ0ODQ0NDQwMDAwLCxUUEhEODQY  
FBQDQAwICAQECAgMDBAUFBg0OERIUFQsLDAwMDA0NDQ0UEhMSEhIRERAQE8ODg4NDAsLCwkJCAg  
GBgUEBAIBAQEBAgIEBAUFBgCHCAgJCgoSLf7jVgEfDg0NDQ4ODg8PDxAQEBERERITExISEhIRERA  
QE8ODg4NDAwLCgoICQcHBQUEBAICAIEBAUFBwcJCAoKCwwMDQ4ODg8QEBARERISEhITAAAAAgA  
AAAADtQP0AAMACgAANYe1IRMzESERMwFKA2z81A/zAWjz/1kmfQHN/p0BYwGeAAAAAUAAAAA/Q  
D9AA/AH8AvwD/Aa8AAAEpDisBLw4/Dx8OBQ8OKwEvDj8Phw4lFQ8OLw49AT8OHw4FFQ8OLw49AT8  
OHw4BHx8zPw09AS8MPQE/DjsBPx01Lx8PHgOFAQECAgQEBQUGBgCHCAgJCAkJCAcIBgcGBQUEAwM  
CAQEBAQIDAwQFBQYHBggHCAkJCAkIBwgHBgYFBQQEAgIB/Z4BAQIDAwQFBQYGBwgHCAkJCAkIBwg  
HBgYFBQDQAwIBAQEBAgMDBAUFBgYHCACICQgJCQgHCAcGBgUFBAMDAgEBvQECAwQEBAYGBgcHCAg  
ICQkICAgHBwcFBgQFAwMCAQECAwMFBAYFBwcHCAgICQkICAgHBwYGBgQEBAWMAf7qAQIDAwUEBgU  
HBwcICAgJCQgICAcHBgYGBAQEAwIBAQIDBAQEBAgYGBwcICAgJCQgICAcHBwUGBAUDAwIB/kQBAGm  
EBgcHCQsLDA00Dw8RERITFBQVfHxYXFxcZGBkZGgkICAgHBwYGBgQEBAWMAQECAwMEBAoEBAMDAgE  
CAgIEBAUFBgYHBwgICAlkDg8NDg0ODA0MDAwLCwsKCQoICQcIBgYGBgQEAWMCAQECAwQEBGwcJCws  
MDQ4PDxEREhMUFBWfHcXFxkYGRkaGhkZGBkXFxcWfHUUFBMSEREPDw4NDAsLCQcHBgQDAgJTCAk  
ICAcHBgYFBQQEAgICAgICBAQFBQYGBwcICAkICQgJBwgGBwYFBQDQAwIBAQEBAgMDBAUFBgcGCAC  
JCAkICQgIBwcGBgUFBACAgICAgIEBAUFBgYHBwgICQgJCAkHCAyHBgUFBAMDAgEBAQECAwMEBQU  
GBwYIBwkI1gkJCAcIBgcGBQUEAwMCAQEBAQIDAwQFBQYHBggHCAkJCAkICAcHBgYFBQQEAgIBAQE  
BAgIEBAUFBgYHBwgICQgJCQgHCAyHBgUFBAMDAgEBAQECAwMEBQUGBwYIBwgJCQgJCAgHBwYGBQU  
EBAICAQEBAQICBAQFBQYGBwcICAn+xhoZGRgZfxcXfHxYVFBQTEhERDw8ODQwLCwkHBwYEAwIBAgI  
CBAQFBQYGBwcICAkICAgIBwcGBgsGBwYIBwgICQkIBwgGBwYFBQDQAwIBAQECAgMEBQUFBgcHCAg  
JCQoKCwoMCwwNDA0NDg0ODg4XFxYWFUUVFBQTEhIRERAPDw4NDQsLCgkIBwYFBAMBAQECAwQGBwc  
JCwsMDQ4PDxEREhMUFBWfHcXFxkYGRkaAgAAAAAD9A01AAgAVAAAArchFSEHFzcnJREVHw4hPw4  
9ASMVIREhFTM9AS8OIQ8OAtV1/k0BSHI/4OD8+AICAwQFBQYHBwcICQkJCQHPCQkJCQgHBwcGBQU  
EAwICXP4xAc9cAgIDBAUFBgCHBwgJCQkJ/jEJCQkJCAcHBwYFBQDQAgICoHRYdD7e3oD9RAkJCAg  
IBwcGBgUEBAMCAQEBAQIDBAQFBgYHBwgICAkJzMwCvMzMCQkICAgHBwYGBgQEAWIBAQEBAgMEBAU  
GBgcHCAgICQADAAAAAAOvA/QAAwBHAF0AAAEIREREHERUfDTMhMz8OES8OIyEjDw0nETMRITUhIw8  
NA1X+DFsCagMEBQUGBgcICAgJCQkB9AkJCQgICAcGBgUFBAMCAQEBAQIDBAUFBgYHCAGICQkJ/gw  
JCQkICAgHBgYFBQDQAgK2WQIT/e0JCQkIBwgHBgYFBAQDAgEC4/2EAnwF/YgJCQgJCAcHBgYGBAQ  
DAGICAgMEBAYGBgcHCAkICQkCeAkJCQgICAcGBgUFAwMDAQEDAwMFBQYGBwgICAkJsv2EAnxbAgI  
DBAUFBgYHCAGICQkAAAAASAN4AAQAAAAAABAAAAAQAQAAAAAQAQAAAAAQAQAAAAAQAQAAAAAQA  
AAAAAwAQABgAAQAAAAAABAAQACgAAQAAAAAABQALADgAAQAAAAAABgAQAEMAAQAQAAAAAACgAsAFM  
AAQAAAAACwASAH8AAwABBAkAAAAACAJEAwABBAkAAQAgAJMAwABBAkAAgAOALMAAwABBAkAAwA  
gAMEAAwABBAkABAAGAOEAwABBAkABQAWAQEAwABBAkABgAgARcAAwABBAkACgBYATcAAwABBAk  
ACwAkAY8gdG9vbGJhcil1tYXRlcmlhbfJlZ3VsYXJ0b29sYmFyLW1hdGVyaWfSDG9vbGJhcil1tYXR  
lcmlhbfZlcnNpb24gMS4wdG9vbGJhcil1tYXRlcmlhbfEzbnQgZ2VuZXJhdGVkIHVzaW5nIFN5bmN  
mdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAAdABvAG8AbABiAGEAcgAtAG0  
AYQB0AGUAcgBpAGEAbABSAGUAZwB1AGwAYQByAHQAbwBvAGwAYgBhAHIALQBtAGEAdABlAHIAaQB  
hAGwAdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABWAGUAcgBzAGkAbwBuACAAMQAuADA  
AdABvAG8AbABiAGEAcgAtAG0AYQB0AGUAcgBpAGEAbABGAG8AbgB0ACAAZwB1AG4AZQByAGEAdAB

```

1AGQAIAB1AHMAaQBAGcAIBTAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcbvACAAUwB0AHU
AZABpAG8AdwB3AHcALgBzAHkAbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAaGAAAAAAAKAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAjAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgE
TARQBFQEWARcBGAEZARoBGwEcAR0BHgEfASABIQEiASMBJAAQVGV4dF9PdXRkZW50XzAwMQtQaWN
0dXJlXzAwMQxTZXR0aW5nc18wMDEQQ29sb3JfcGlja2VyXzAwMhBBbGlnbl19DZW50ZXJfMDA2CEX
pbmVfMDAxDVVuZGVybGluZV8wMDEMU29ydf9aLUFFMDAxCFVuZG9fMDAxEENoYXJ0X2J1YmJsZV8
wMDELRG93bmXvYWRfMDAPVGv4dF9pbmRlbnRfMDAxEkNoYXJ0X0RvdWdobnV0XzAwMQlDbGVhc18
wMDINTnVtYmVyaW5nXzAwMQxTb3J0X0EtWl8wMDEKSXRhbG1jXzAwMQtCdWxsZXRzXzAwMQlQYXN
0ZV8wMDEIUmVkb18wMDEPQ2hhcnRfcmlkYXJfMDAxD0FsaWduX1JpZ2h0XzAwMQlUYWJsZV8wMDE
OQWxpZ25fTGVMdF8wMDEITWVudV8wMDEHQ3V0XzAwMghCb2xkXzAwMRFBbGlnbl19KdXN0aWZ5XzA
wMQpSZWxvYWRfMDAxClNlYXJjaF8wMDEKVXBsb2FkXzAwMQpEZXRnpZ25fMDA1CkV4cG9ydf8wMDE
IQ29weV8wMDIAAA==) format('truetype');
 font-weight: normal;
 font-style: normal;
}
.e-bigger .e-tbar-btn .tb-icons {
 font-size: 18px;
}
.e-tbar-btn .tb-icons {
 font-family: 'Material_toolbar';
 speak: none;
 font-size: 16px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
}
.e-tbar-btn .tb-icons.e-numbering-icon {
 font-size: 17px;
}
.e-cut-icon:before {
 content: "\e719"
}
.e-copy-icon:before {
 content: "\e721"
}
.e-paste-icon:before {
 content: "\e712"
}
.e-color-icon:before {
 content: "\e703";
}
.e-bold-icon:before {
 content: "\e71a"
}
.e-underline-icon:before {
 content: "\e706";
}
.e-alignleft-icon:before {
 content: "\e717"
}
.e-alignright-icon:before {
 content: "\e715"
}
.e-aligncenter-icon:before {
 content: "\e704"
}
}

```

```
.e-alignjustify-icon:before {
 content: "\e71b"
}
.e-upload-icon:before {
 content: "\e71e"
}
.e-download-icon:before {
 content: "\e70a"
}
.e-indent-icon:before {
 content: "\e70b"
}
.e-outdent-icon:before {
 content: "\e700"
}
.e-clear-icon:before {
 content: "\e70d"
}
.e-reload-icon:before {
 content: "\e71c"
}
.e-export-icon:before {
 content: "\e720";
}
.e-italic-icon:before {
 content: "\e710"
}
.e-bullets-icon:before {
 content: "\e711";
}
.e-numbering-icon:before {
 content: "\e70e";
}
.e-ascending-icon:before {
 content: "\e70f";
}
.e-descending-icon:before {
 content: "\e707";
}
.control-wrapper {
 flex-direction: inherit
}
.e-sample-resize-container {
 height: 42px;
}
@@media only screen and (min-width: 1224px) {
 .e-sample-resize-container {
 width: 75%;
 }
}
@@media only screen and (min-width: 1824px) {
 .e-sample-resize-container {
 width: 50%;
 }
}
.material .e-bigger .e-toolbar .e-tbar-btn .e-icons {
 font-size: 18px;
```

```

 }
 .material .e-toolbar .e-tbar-btn .e-icons {
 font-size: 16px;
 }
</style>

```

### SCROLLSTEP.CS

```

public ActionResult Index()
{
 return View();
}

```

## ToolTip

### Getting Started with ASP.NET MVC Tooltip Control

This section briefly explains about how to include [ASP.NET MVC Tooltip](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://www.nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Tooltip control

Now, add the Syncfusion ASP.NET MVC Tooltip control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().Tooltip("Tooltip").Content("Lets go green & Save Earth
!!!").ContentTemplate(@Show Tooltip).Render()
```

Press **Ctrl+F5** (Windows) or **⌘+F5** (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Tooltip control will be rendered in the default web browser.

Lets go green & Save Earth !!!

Show Tooltip

### Initialize Tooltip within a container

You can create Tooltips on multiple targets within a container. To do so, you have to define specific target elements to the [Target](#) property so that the Tooltip is initialized only on matched targets within a container. In this case, the Tooltip content is assigned from the `title` attribute of the target element.

Refer to the following code example to create a Tooltip on multiple targets within a container.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("Tooltip").Target(".e-info").Position(Syncfusion.EJ2.Popups.Position.RightCenter).ContentTemplate(
@<form id="details" role="form">
 <table>
 <tr>
 <td class="info"> User Name:</td>
 <td> <input type="text" class="e-info"
name="firstname" title="Please enter your name"> </td>
 </tr>
 <tr>
 <td class="info"> Email Address:</td>
 <td> <input type="text" class="e-info"
name="email" title="Enter a valid email address"></td>
 </tr>
 <tr>
 <td class="info"> Password:</td>
 <td> <input type="password" class="e-info"
name="password" title="Be at least 8 characters length"></td>
 </tr>
 <tr>
 <td class="info"> Confirm Password:</td>
 <td> <input type="password" class="e-info"
name="Cpwd" title="Re-enter your password"></td>
 </tr>
 </table>
 <input id="sample" type="submit" value="Submit">
 <input id="clear" type="reset" value="Reset">
</form>).Render()

<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 #details table th, #details table td {
 padding: 15px;
 text-align: left;
 }
 #details .info {
 font-weight: bold;
```

```

 }
</style>

```

User Name  Please enter your name

Email Address

Password

Confirm Password

**Note:** In the above sample, `#details` refers to the container's id, and the target `.e-info` refers to the target elements available within that container.

**Note:** [View Sample in GitHub.](#)

See also

- [Positioning Tooltip](#)
- [Tooltip Open Mode](#)
- [Customize the Tooltip](#)

## Setting Dimension in Tooltip Control

### Height and width

The Tooltip can either be assigned auto height and width values or specific pixel values. The `width` and `height` properties are used to set the outer dimension of the Tooltip element. The default value for both the properties is `auto`. It also accepts string and number values in pixels.

The following sample explains how to set dimensions for the Tooltip.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

@Html.EJS().Tooltip("tooltip").Width("150px").Height("40px").Content("Tooltip
with specific width and height").ContentTemplate(@<div>
 <div>
 Show Tooltip
 </div>
</div>).Render()
<style>
 #tooltip {
 position: absolute;
 left: calc(50% - 60px);

```



```

 top: 38%;
 }
</style>

```

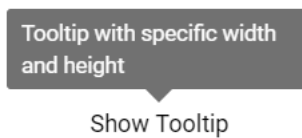
### HEIGHT-WIDTH.CS

```

public ActionResult Dimension()
{
 return View();
}

```

Output be like the below.



### Scroll mode

When **height** is specified with a certain pixel value and the Tooltip content overflows, the scrolling mode gets enabled.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

@Html.EJS().ToolTip("tooltipContent").Width("300px").Height("60px").Target("#target").Content("Environmentally friendly or environment-friendly, (also referred to as eco-friendly, nature-friendly, and green) are marketing and sustainability terms referring to goods and services, laws, guidelines and policies that inflict reduced, minimal, or no harm upon ecosystems or the environment.").IsSticky(true).ContentTemplate(@<div>
 <div id='container'>
 <p>
 A green home is a type of house designed to be

 <u>environmentally friendly</u>
 and sustainable. And also focuses on the efficient use
of "energy, water, and building materials." As green homes
 have become more prevalent we have also seen the emergence
of green affordable housing.
 </p>
 </div>
</div>).Render()

```

### SCROLL-MODE.CS

```

public ActionResult Dimension()
{
 return View();
}

```

```
}

```

Output be like the below.



**Note:** The scrolling mode can best be seen when the sticky mode of the Tooltip is enabled. To enable sticky mode, set the `isSticky` property to true.

## Content in Tooltip Control

A text or a piece of information assigned to the Tooltip's `content` property will be displayed as the main text stream of the Tooltip. It can be a string or a template content. If the `content` property is not provided with any specific value, then it takes the value assigned to the `title` attribute of the target element on which the Tooltip was initialized. The content can also dynamically be assigned to the Tooltip via Fetch.

### Template content

Any text or image can be added to the Tooltip, by default. To customize the Tooltip layout or to create your own visualized element on the Tooltip, template can be used.

Refer to the following code example to add formatted HTML content to the Tooltip.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

@Html.EJS().Tooltip("tooltipContent").Content("<div><p>Environmental
ly friendly or environment-friendly, <i>(also
referred to as eco-friendly, nature-friendly, and green)</i> are marketing
and sustainability terms referring to goods and services, laws, guidelines
and policies that inflict reduced, minimal, or no harm upon ecosystems or
the environment.</p></div>").Target("#target").ContentTemplate(@<div>
 <div id='container'>
 <p>
 A green home is a type of house designed to be

 <u>environmentally friendly</u>
 and sustainable. And also focuses on the efficient use
of "energy, water, and building materials." As green homes
 have become more prevalent we have also seen the emergence
of green affordable housing.
 </p>
 </div>
</div>).Render()
```

### TEMPLATE.CS

```
public ActionResult Template()
{
 return View();
}
```

Output be like the below.

A green home is a type of house designed to be environmentally friendly and sustainable. And also focuses on the efficient use of "energy, water and building materials". It is more houses have become more prevalent we have also seen th

Environmentally friendly or environment-friendly, (also referred to as eco-friendly, nature-friendly, and green) are marketing and sustainability terms referring to goods and services, laws, guidelines and policies that inflict reduced, minimal, or no harm upon ecosystems or the environment.

### Dynamic content via Fetch

The Tooltip content can be dynamically loaded by making use of the Fetch call. The Fetch request is usually made within the `beforeRender` event of the Tooltip, and then the Tooltip's `content` is assigned the value retrieved on it's success.

**Note:** The Tooltip **target** property includes a unique identifier used to associate Tooltips with specific elements on a webpage or application interface. When setting the Tooltip **target** value as a GUID (Globally Unique Identifier), it's important to note that the GUID must start with a combination of **letters** before the numeric portion of the GUID. For example, **target: '# + ' tooltip' + '96ad88bd-294c-47c3-999b-a9daa3285a05'**.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div id='container'>
 <h4 class="list-header">National Sports</h4>
 <!-- Tooltip element rendering for AjaxContent loading -->

@Html.EJS().Tooltip("Tooltip").Content("Loading...").Target("#countrylist
[title]").Position(Syncfusion.EJ2.Popups.Position.RightCenter).BeforeRender(
"onBeforeRender").ContentTemplate(@<div>

@Html.EJS().ListView("countrylist").Enable(true).DataSource((IEnumerable<obj
ect>)ViewBag.data).Fields(new ListViewFieldSettings { Text = "text", Tooltip
= "id" }).Render()
</div>).Render()
</div>
<script>
/**
 * Process tooltip ajax content.
 */
function onBeforeRender(args) {
 var _this = this;
 this.content = 'Loading...';
 this.dataBind();
 var fetchApi = new
ej.base.Fetch("@Url.Content("~/Scripts/tooltip/tooltipdata.json")", 'GET');
 fetchApi.send().then(function (result) {
 for (var i = 0; i < result.length; i++) {
 if (result[i].Id === args.target.getAttribute('data-
content')) {
 _this.content = "<div class='contentWrap'><span class="
+ result[i].Class + "><div class='def'>" + result[i].Sports +
"</div></div>";
 }
 }
 });
}
```

```

 }
 _this.dataBind();
 }, function (reason) {
 _this.content = reason.message;
 _this.dataBind();
 });
}
</script>
<link href="../../Content/tooltip/icons.css" rel="stylesheet" />
<style>
 .contentWrap {
 padding: 3px 0;
 line-height: 16px;
 }
 .e-bigger [class^="sports-icon-"],
 .e-bigger [class*=" sports-icon-"] {
 font-size: 18px;
 }
 [class^="sports-icon-"],
 [class*=" sports-icon-"] {
 font-family: 'sportsicons';
 speak: none;
 font-size: 16px;
 font-style: normal;
 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 padding-right: 6px;
 vertical-align: middle;
 }
 .sports-icon-cricket:before {
 content: "\e703";
 }
 .sports-icon-archery:before {
 content: "\e705";
 }
 .sports-icon-table-tennis:before {
 content: "\e702";
 }
 .sports-icon-baseball:before {
 content: "\e706";
 }
 .sports-icon-hockey:before {
 content: "\e701";
 }
 .sports-icon-shooting:before {
 content: "\e700";
 }
 .def {
 float: right;
 }
 #countrylist {
 border: 1px solid #dddddd;
 border-radius: 3px;
 max-width: 170px;
 margin: 0 auto;
 overflow: hidden;

```

```

 }
 .list-header {
 text-align: center;
 color: rgba(0, 0, 0, 0.54);
 }
 @@media (max-width: 481px) {
 #countrylist {
 margin: 0;
 }
 .list-header {
 text-align: left;
 }
 }
}
</style>

```

### AJAXCONTENT.CS

```

public ActionResult AjaxContent()
{
 List<object> country = new List<object>();
 country.Add(new { id = "1", text = "Australia" });
 country.Add(new { id = "2", text = "Bhutan" });
 country.Add(new { id = "3", text = "China" });
 country.Add(new { id = "4", text = "Cuba" });
 country.Add(new { id = "5", text = "India" });
 country.Add(new { id = "6", text = "Switzerland" });
 country.Add(new { id = "7", text = "United States" });
 ViewBag.data = country;
 return View();
}

```

Output be like the below.



### ToolTip Positioning in Tooltip Control

Tooltips can be attached to 12 static locations around the target. On initializing the Tooltip, you can set the position property with any one of the following values:

- TopLeft
- TopCenter
- TopRight
- BottomLeft
- BottomCenter

- BottomRight
- LeftTop
- LeftCenter
- LeftBottom
- RightTop
- RightCenter
- RightBottom

**Note:** By default, Tooltip is placed at the TopCenter of the target element.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div class="control-section">
 @Html.EJS().Tooltip("Tooltip").Content("Tooltip
content").Position(Syncfusion.EJ2.Popups.Position.TopCenter).ContentTemplate
(@<div>
 <div id='container'>
 <div id="tooltip" title="Tooltip content">
 Show tooltip
 <select id="positions" class="form-control"
onchange="onChange(this)">
 <option value="TopLeft">Top Left</option>
 <option value="TopCenter" selected="">Top
Center</option>
 <option value="TopRight">Top Right</option>
 <option value="BottomLeft">Bottom Left</option>
 <option value="BottomCenter">Bottom Center</option>
 <option value="BottomRight">Bottom Right</option>
 <option value="LeftTop">Left Top</option>
 <option value="LeftCenter">Left Center</option>
 <option value="LeftBottom">Left Bottom</option>
 <option value="RightTop">Right Top</option>
 <option value="RightCenter">Right Center</option>
 <option value="RightBottom">Right Bottom</option>
 </select>
 </div>
 </div>
</div>).Render()
</div>
<style>
 #target {
 background-color: #ecec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 70px auto;
 padding: 20px;
 width: 200px;
 }
 /* csslint ignore:start */
 .customtip.e-tooltip-wrap {
 padding: 4px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
```

```

 height: 20px;
 width: 12px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
 height: 20px;
 width: 12px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
 height: 12px;
 width: 20px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
 height: 12px;
 width: 20px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
 border-left: 6px solid transparent;
 border-right: 6px solid transparent;
 border-top: 20px solid #616161;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
 border-bottom: 20px solid #616161;
 border-left: 6px solid transparent;
 border-right: 6px solid transparent;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
 border-bottom: 6px solid transparent;
 border-right: 20px solid #616161;
 border-top: 6px solid transparent;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
 border-bottom: 6px solid transparent;
 border-left: 20px solid #616161;
 border-top: 6px solid transparent;
 }
}
</style>
<script>
 var tooltip;
 window.onload = function () {
 tooltip = document.getElementById('Tooltip').ej2_instances[0];
 }
 function onChange(args) {
 tooltip.position = args.value;
 }
</script>

```

### TOOLTIPPOSITIONS.CS

```

public ActionResult TooltipAnimation()
{
 ViewBag.content = "Tooltip Content";
 return View();
}

```

Output be like the below.



### Tip pointer positioning

The Tooltip pointer can be attached or detached from the Tooltip by using the `showTipPointer` property. Pointer positions can be adjusted using the `tipPointerPosition` property that can be assigned to one of the following values:

- `auto`
- `start`
- `middle`
- `end`

The following code example illustrates how to set the pointer to the start position of the Tooltip.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("Tooltip").Content("Tooltip
content").TipPointerPosition(Syncfusion.EJ2.Popups.TipPointerPosition.Start)
.ContentTemplate(@<div>
 <div id='target'>
 Show Tooltip
 </div>
</div>).Render()
<style>
 #container {
 visibility: hidden;
 }
 #loader {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 50px auto;
 padding: 20px;
 width: 250px;
 }
}
```

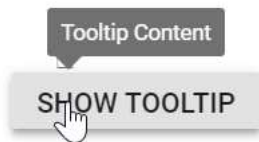


```
</style>
```

### TIPPOINTER.CS

```
public ActionResult TipPointer()
{
 ViewBag.content = "Tooltip Content";
 return View();
}
```

Output be like the below.



By default, tip pointers are auto adjusted so that the arrow does not point outside the target element.

### Dynamic positioning

The Tooltip and its tip pointer can be positioned dynamically based on the target's location. This can be achieved by using the `refresh` method, which auto adjusts the Tooltip over the target.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div class="control-section">
 <!-- Tooltip element -->
 @Html.EJS().Tooltip("targetContainer").Content("Drag
me").Target("#demoSmart").ContentTemplate(@<div id="demoSmart">
 </div>).Render()
 </div>
<script type="text/javascript">
 var tooltip, ele;
 window.onload = function () {
 ele = document.getElementById('demoSmart');
 tooltip =
document.getElementById("targetContainer").ej2_instances[0];
 <!-- Initialize Draggable for tooltip element -->
 var drag = new ej.base.Draggable(ele, {
 clone: false,
 dragArea: '#targetContainer',
 drag: function (args) {
 if (args.element.getAttribute('data-tooltip-id') === null) {
 tooltip.open(args.element);
 }
 else {
 tooltip.refresh(args.element);
 }
 },
 dragStart: function (args) {
 if (args.element.getAttribute('data-tooltip-id') !== null) {
```

```

 return;
 }
 tooltip.open(args.element);
},
dragStop: function (args) {
 tooltip.close();
}
});
}
</script>
<style type="text/css">
 #demoSmart {
 background: rebeccapurple;
 height: 50px;
 left: 35%;
 position: absolute;
 top: 35%;
 width: 50px;
 }
 #targetContainer {
 margin: 10px;
 min-height: 320px;
 width: 100%;
 float: left;
 border: 1px solid #dddddd;
 border-radius: 3px;
 }
 .control-section {
 padding: 3px;
 margin-right: 20px;
 }
</style>

```

### DYNAMICPOSITION.CS

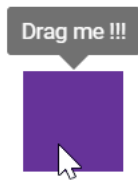
```

public ActionResult DynamicPosition()
{
 ViewBag.custom = "Drag Me";
 return View();
}

```

**Note:** When mouse trailing option is enabled, the tip pointer position gets auto adjusted based on the target, and other position values like start, end, and middle are not applied (to prevent the pointer from moving out of target).

Output be like the below.



### Mouse Trailing

Tooltips can be positioned relative to the mouse pointer. This behavior can be enabled or disabled by using the `mouseTrail` property. By default, it is set to `false`.

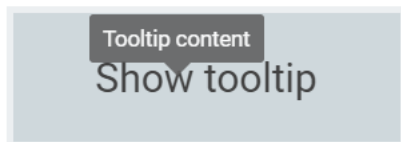
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("Tooltip").MouseTrail(true).Content("Tooltip
content").ContentTemplate(@<div>
 <div id="target" style="margin: 50px;">
 Show tooltip
 </div>
</div>).Render()
<style>
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 80px auto;
 padding: 20px;
 width: 200px;
 }
</style>
```

### MOUSETRILING.CS

```
public ActionResult MouseTrailing()
{
 return View();
}
```

Output be like the below.



**Note:** When mouse trailing option is enabled, the tip pointer position gets auto adjusted based on the target, and other position values like start, end, and middle are not applied (to prevent the pointer from moving out of target).

### Setting offset values

Offset values are set to specify the distance between the target and tooltip element.

`offsetX` and `offsetY` properties are used to specify the offset left and top values.

- `offsetX` specifies the distance between the target and Tooltip element in X axis.
- `offsetY` specifies the distance between the target and Tooltip element in Y axis.

The following code example illustrates how to set offset values.

#### CSHTML

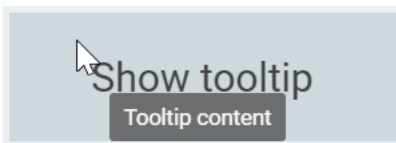
```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

@Html.EJS().ToolTip("tooltip").OffsetX(30).OffsetY(30).MouseTrail(true).Show
TipPointer(false).Content("Tooltip content").ContentTemplate(@<div>
 <div id="target">Show tooltip</div>
</div>).Render()
<style>
 #target {
 background-color: #ecec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 80px auto;
 padding: 20px;
 width: 200px;
 }
</style>
```

#### OFFSETVALUES.CS

```
public ActionResult offsetvalues()
{
 ViewBag.content = "Tooltip Content";
 return View();
}
```

Output be like the below.



**Note:** By default, collision is handled automatically and therefore when collision is detected the Tooltip fits horizontally and flips vertically.

### Open Mode in Tooltip Control

You can decide the mode on which the Tooltip is to be opened on a page, i.e., on hovering, focusing, or clicking on the target elements.

**Note:** On mobile devices, Tooltips appear when you tap and hold the element, even if the `opensOn` option is assigned with `Hover`.

<br/> Tooltips are also displayed as long as you continue to tap and hold the element. On lift, it disappears in the next 1.5 seconds.

<br/> If there is another action before that time ends, then the Tooltip disappears.

The `opensOn` property can take either a single or a combination of multiple values, separated by space from the following options. The table below explains how the Tooltip opens on both desktop and mobile based on the value(s) assigned to the `opensOn` property. By default, it takes `Auto` value.

| Values              | Desktop                                                                                                                                                     | Mobile                                                   |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|
| <code>Auto</code>   | Tooltip appears when you hover over the target or when the target element receives the focus.                                                               | Tooltip opens on tap and hold of the target element.     |
| <code>Hover</code>  | Tooltip appears when you hover over the target.                                                                                                             | Tooltip opens on tap and hold of the target element.     |
| <code>Click</code>  | Tooltip appears when you click a target element.                                                                                                            | Tooltip appears when you single tap the target element.  |
| <code>Focus</code>  | Tooltip appears when you focus (say through tab key) on a target element.                                                                                   | Tooltip appears with a single tap on the target element. |
| <code>Custom</code> | Tooltip is not triggered by any default action. So, you have to bind your own events and use either <code>open</code> or <code>close</code> public methods. | Same as Desktop.                                         |

To open the Tooltip for multiple actions, say while hovering over or clicking on a target element, the `opensOn` property can be assigned with multiple values, separated by space as `Hover Click`.

**Note:** `Auto` value cannot be used with any combination for multiple values.

The following code example shows how to set the open mode for Tooltips.

#### CSHTML

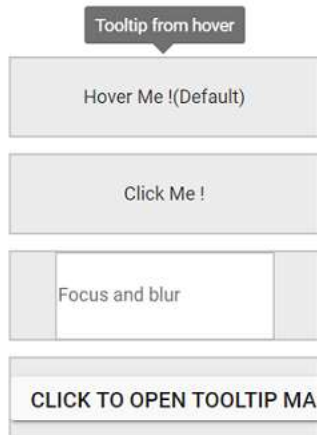
```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("hover").OpensOn("Hover").Content("Tooltip from
hover").ContentTemplate(@<div id="tooltiphover" class="blocks">Hover
Me !(Default)</div>).Render()
@Html.EJS().Tooltip("click").OpensOn("Click").Content("Tooltip from
Click").ContentTemplate(@<div id="tooltipclick" class="blocks">Click
Me !</div>).Render()
@Html.EJS().Tooltip("focus").OpensOn("Focus").Content("Tooltip from
focus").Target(".e-info").ContentTemplate(@<div class="blocks"><input
class="e-info" id="tooltipfocus" type="text" placeholder="Focus and blur"
/></div>).Render()
@Html.EJS().Tooltip("custom").OpensOn("Custom").Content("Tooltip from
custom mode").ContentTemplate(@<div id="tooltipcustom" class="blocks">
@Html.EJS().Button("open").Content("Click to open tooltip
manually").Render()
</div>).Render()
<style>
```

```
#hover, #click, #focus, #custom, #open {
 width: 200px;
}
.blocks {
 background-color: #ecec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 display: inline-block;
 line-height: 50px;
 margin: 0 10px 10px 0;
 overflow: hidden;
 text-align: center;
 vertical-align: middle;
 width: 200px;
}
</style>
<script>
 document.getElementById('open').addEventListener("click", function () {
 var customTooltip =
document.getElementById('custom').ej2_instances[0];
 if (this.getAttribute("data-tooltip-id")) {
 customTooltip.close();
 } else {
 customTooltip.open(this);
 }
 });
</script>
```

### OPEN-MODE.CS

```
public ActionResult OpenMode()
{
 ViewBag.hover = "Tooltip from hover";
 ViewBag.click = "Tooltip from click";
 ViewBag.focus = "Tooltip from focus";
 ViewBag.custom = "Tooltip from custom mode";
 return View();
}
```

Output be like the below.



### Custom open mode

Other than the above specified options, the Custom mode makes the Tooltip appear on screen for user-defined custom actions such as right-click, double-click, and so on. Here, the tooltip is not triggered by any default action, and you have to bind your own events and use either `open` or `close` public methods to show or hide the Tooltips.

The following code example shows how to define custom open mode for the Tooltip.

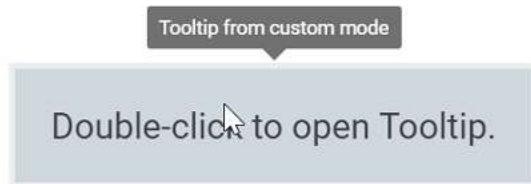
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("tooltip").OpensOn("Custom").Content("Tooltip from
custom mode").ContentTemplate(@<div id="box">
 Double-click to open Tooltip.
</div>).Created("created").Render()
<style>
 #box {
 background-color: #ecec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 50px auto;
 padding: 20px;
 width: 250px;
 }
</style>
<script>
 function created() {
 document.getElementById('box').addEventListener("dblclick", function
() {
 var tooltip =
document.getElementById('tooltip').ej2_instances[0];
 if (this.getAttribute("data-tooltip-id")) {
 tooltip.close();
 } else {
 tooltip.open(this);
 }
 });
 }
</script>
```

**CUSTOM-MODE.CS**

```
public ActionResult CustomMode()
{
 return View();
}
```

Output be like the below.

**Sticky mode**

With this mode set to **true**, Tooltips can be made to show up on the screen as long as the close icon is pressed. In this mode, close icon is attached to the Tooltip located at the top right corner. This mode can be enabled or disabled using the **isSticky** property.

**CSHTML**

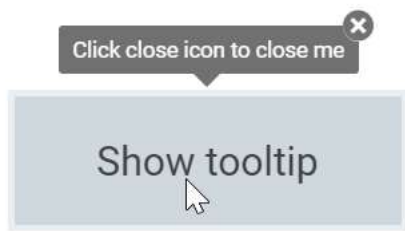
```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("tooltip").IsSticky(true).Content("Click close icon
to close me").ContentTemplate(@<div id="target">
 Show tooltip
</div>).Render()
<style>
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 80px auto;
 padding: 20px;
 width: 200px;
 }
</style>
```

**STICKY-MODE.CS**

```
public ActionResult StickyMode()
{
 ViewBag.content = "Click close icon to close me";
 return View();
}
```

Output be like the below.





### Open/Close Tooltip with delay

The Tooltips can be opened or closed after some delay by using the `openDelay` and `closeDelay` properties.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

@Html.EJS().Tooltip("tooltip").OpenDelay(1000).CloseDelay(1000).Content("Too
ltip with delay").ContentTemplate(@<div id="target">
 Show tooltip
</div>).Render()
<style>
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 80px auto;
 padding: 20px;
 width: 200px;
 }
</style>
```

#### DELAY-MODE.CS

```
public ActionResult DelayMode()
{
 ViewBag.content = "Tooltip with delay";
 return View();
}
```

Output be like the below.



## Animation in Tooltip Control

To animate the Tooltip, a set of specific animation effects are available, and it can be controlled using the `animation` property. The animation property also allows you to set delay, duration, and various other effects of your choice.

`AnimationModel` is derived from base to apply the chosen animation effect, duration, etc. on Tooltips.

By default, Tooltip entrance occurs over 150 ms using the `ease-out` timing function. It exits also at 150 ms, but uses `ease-in` timing function.

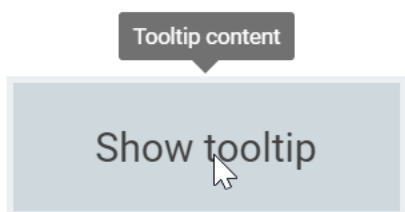
### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("tooltip").Content("Tooltip content").Animation(new
{
 open = new { effect = "ZoomIn", duration = 1000, delay = 0 },
 close = new { effect = "ZoomOut", duration = 500, delay = 0 }
}).ContentTemplate(@<div id="target">
 Show tooltip
</div>).Render()
<style>
 #target {
 background-color: #ecec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 50px auto;
 padding: 10px;
 width: 100px;
 }
</style>
```

### ANIMATION.CS

```
public ActionResult TooltipAnimation()
{
 ViewBag.animation = new
 {
 open = new { effect = "ZoomIn", duration = 1000, delay = 0 },
 close = new { effect = "ZoomOut", duration = 500, delay = 0 }
 };
 return View();
}
```

Output be like the below.



The default animation effect for the Tooltip is set to `FadeIn` for its open action, and `FadeOut` for its close action. The default `duration` is set to 150 ms and `delay` is set to 0.

#### Animation effects

The animation effects that are applicable to Tooltips are:

- `FadeIn`
- `FadeOut`
- `FadeZoomIn`
- `FadeZoomOut`
- `FlipXDownIn`
- `FlipXDownOut`
- `FlipXUpIn`
- `FlipXUpOut`
- `FlipYLeftIn`
- `FlipYLeftOut`
- `FlipYRightIn`
- `FlipYRightOut`
- `ZoomIn`
- `ZoomOut`
- `None`

When the `effect` is specified as `None`, no effect will be applied to the Tooltip, and animation is considered to be set to `off`.

**Note:** Some of the above animation effects suits the Tooltip better when its tip pointer is hidden.

<br/> This can be achieved by setting the `showTipPointer` to false.

#### Animating via open/close methods

Animations can also be applied on Tooltips dynamically through `open` and `close` methods. These methods accept the animation model as an optional parameter. If you pass `TooltipAnimationSettings`, animation takes this model; otherwise, it takes the values from the `animation` property. It is also possible to pass different animations for Tooltips on each target.

Refer to the code snippet below to apply animations using public methods.

#### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().ToolTip("tooltip").Content("Tooltip
content").OpensOn("Custom").ContentTemplate(@<div id="target">
 Show tooltip
</div>).Created("created").Render()
<style>
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 50px auto;
```

```

padding: 10px;
width: 100px;
}
</style>
<script>
function created() {
document.getElementById('target').addEventListener("click", function ()
{
var tooltip =
document.getElementById('tooltip').ej2_instances[0];
if (this.getAttribute("data-tooltip-id")) {
var closeAnimation = { effect: 'FadeOut', duration: 1000 }
tooltip.close(closeAnimation);
} else {
var openAnimation = { effect: 'FadeIn', duration: 1000 }
tooltip.open(this, openAnimation);
}
});
}
</script>

```

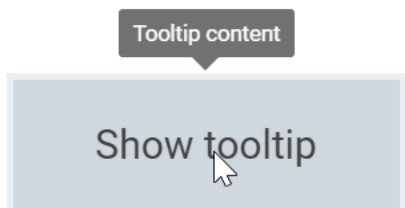
### ANIMATION-OPENCLOSE.CS

```

public ActionResult TooltipAnimation()
{
return View();
}

```

Output be like the below.



### Apply transition

The transition effect can be applied on Tooltips by using the `beforeRender` event as given in the following work-around code example.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

@Html.EJS().Tooltip("tooltip").Target(".circletool").BeforeRender("onBeforeRender").AfterClose("onAfterClose").CloseDelay(1000).Animation(new {
 open = new { effect = "ZoomIn", duration = 500 },
 close = new { effect = "ZoomOut", duration = 500 }
}).ContentTemplate(@<div>
 <h3> Transition effect </h3>
 <div id="box" class="e-prevent-select">

```

```

 <div class="circletool" style="top:18%;left:5%" title="This is
Turtle !!!"></div>
 <div class="circletool" style="top:30%;left:30%" title="This is
Snake !!!"></div>
 <div class="circletool" style="top:80%;left:80%" title="This is
Croc !!!"></div>
 <div class="circletool" style="top:65%;left:50%" title="This is
String Ray !!!"></div>
 <div class="circletool" style="top:75%;left:15%" title="This is
Blob Fish !!!"></div>
 <div class="circletool" style="top:30%;left:70%" title="This is
Mammoth !!!"></div>
 </div>).Render()
<style>
 #box {
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 height: 200px;
 margin-left: 10px;
 margin-right: 10px;
 position: relative;
 }
 .circletool {
 background: yellowgreen;
 border-radius: 50px;
 height: 20px;
 position: absolute;
 width: 20px;
 }
</style>
<script>
 function onBeforeRender(args) {
 if (args.element) {
 // here prevent animation while apply transition
 this.animation = { open: { effect: 'None' } };
 args.element.style.display = 'block';
 args.element.style.transitionProperty = 'left,top';
 args.element.style.transitionDuration = '1000ms';
 }
 }
 function onAfterClose(args) {
 // restore the animation effects
 this.animation = { open: { effect: 'ZoomIn', duration: 500 }, close:
{ effect: 'ZoomOut', duration: 500 } };
 }
</script>

```

### ANIMATION-TRANSITION.CS

```

public ActionResult TooltipAnimation()
{
 ViewBag.animation = new
 {
 open = new { effect = "ZoomIn", duration = 500 },
 close = new { effect = "ZoomOut", duration = 500 }
 }
}

```

```
};
return View();
}
```

Output be like the below.

Transition effect



## Customization in Tooltip Control

The Tooltip can be customized by using the `cssClass` property, which accepts custom CSS class names that define specific user-defined styles and themes to be applied on the Tooltip element.

### Tip pointer customization

Styling the tip pointer's size, background, and border color's can be done using the `cssClass` property, as given below.

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@Html.EJS().Tooltip("tooltip").CssClass("customtip").Content("Tooltip
arrow customized").ContentTemplate(@<div id="target">
 Show tooltip
</div>).Render()
<style>
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 70px auto;
 padding: 20px;
 width: 200px;
 }
 /* csslint ignore:start */
 .customtip.e-tooltip-wrap {
 padding: 4px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
 height: 20px;
 width: 12px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
 height: 20px;
 width: 12px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
 height: 12px;
 width: 20px;
 }
```

```

 }
 .customtip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
 height: 12px;
 width: 20px;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
 border-left: 6px solid transparent;
 border-right: 6px solid transparent;
 border-top: 20px solid #616161;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
 border-bottom: 20px solid #616161;
 border-left: 6px solid transparent;
 border-right: 6px solid transparent;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
 border-bottom: 6px solid transparent;
 border-right: 20px solid #616161;
 border-top: 6px solid transparent;
 }
 .customtip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
 border-bottom: 6px solid transparent;
 border-left: 20px solid #616161;
 border-top: 6px solid transparent;
 }
}
</style>

```

### TOOLTIP-POINTER.CS

```

public ActionResult TooltipPointer()
{
 return View();
}

```

Output be like the below.



### Tooltip customization

The complete look and feel of the Tooltip can be customized by changing its background color, opacity, content font, etc. The following code example shows the way to achieve it.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups

```

```

@Html.EJS().Tooltip("target").CssClass("customtooltip").Content("Tooltip
customized").ContentTemplate(@<div id="content">
 Show tooltip
</div>).Render()
<style>
 #target {
 background-color: #ececec;
 border: 1px solid #c8c8c8;
 box-sizing: border-box;
 margin: 70px auto;
 padding: 20px;
 width: 200px;
 }
 /* csslint ignore:start */
 .customtooltip.e-tooltip-wrap .e-tip-content {
 line-height: 20px;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
 height: 12px;
 left: 50%;
 top: 100%;
 width: 24px;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
 height: 12px;
 left: 50%;
 top: -9px;
 width: 24px;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
 height: 24px;
 left: -9px;
 top: 48%;
 width: 12px;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
 height: 24px;
 left: 100%;
 top: 50%;
 width: 12px;
 }
 .customtooltip.e-tooltip-wrap {
 border-radius: 4px;
 opacity: 1;
 }
 .customtooltip.e-tooltip-wrap.e-popup {
 background-color: #fff;
 border: 2px solid #000;
 }
 .customtooltip.e-tooltip-wrap .e-tip-content {
 color: #000;
 font-size: 12px;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
 border-left: 12px solid transparent;
 border-right: 14px solid transparent;

```



```

 border-top: 12px solid #000;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
 border-bottom: 12px solid #000;
 border-left: 12px solid transparent;
 border-right: 12px solid transparent;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
 border-bottom: 12px solid transparent;
 border-right: 12px solid #000;
 border-top: 12px solid transparent;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
 border-bottom: 12px solid transparent;
 border-left: 12px solid #000;
 border-top: 12px solid transparent;
 }
 .customtooltip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
 color: #fff;
 font-size: 25.9px;
 }
}
</style>

```

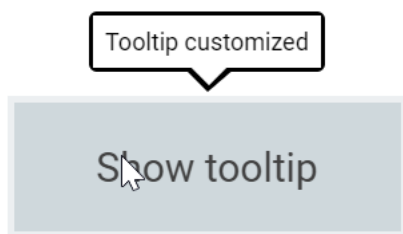
### TOOLTIP-VIEW.CS

```

public ActionResult TooltipView()
{
 return View();
}

```

Output be like the below.



### Styles and Appearance

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the tooltip

Use the following CSS to customize the tooltip.

```

`css
.e-tooltip-wrap {
border-radius: 4px;
opacity: 1;

```

```
}
,
```

### Customizing the tooltip popup

Use the following CSS to customize the tooltip popup properties.

```
`css

.e-tooltip-wrap.e-popup {
background-color: #fff;
border: 2px solid #000;
}
,
```

### Customizing the tooltip content

Use the following CSS to customize the tooltip content.

```
`css

.e-tooltip-wrap .e-tip-content {
color: red;
font-size: 12px;
line-height: 20px;
}
,
```

### Customizing the tooltip arrow tip

Use the following CSS to customize the tooltip arrow tip.

```
`css

/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
height: 12px;
left: 50%;
top: 100%;
width: 24px;
}

/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip.e-tip-top {
height: 12px;
left: 50%;
top: -9px;
```

```
width: 24px;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip.e-tip-left {
height: 24px;
left: -9px;
top: 48%;
width: 12px;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip.e-tip-right {
height: 24px;
left: 100%;
top: 50%;
width: 12px;
}
`
```

#### Customizing the tooltip inner tip

Use the following CSS to customize the tooltip inner tip.

```
`css
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-right,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-left,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom,
.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top {
color: #fff;
font-size: 25.9px;
}
`
```

#### Customizing the tooltip outer tip

Use the following CSS to customize the tooltip outer tip.

```
`css
/ To customize the arrow tip at bottom /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
border-left: 12px solid transparent;
}
```

```
border-right: 14px solid transparent;
border-top: 12px solid #000;
}
/ To customize the arrow tip at top /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
border-bottom: 12px solid #000;
border-left: 12px solid transparent;
border-right: 12px solid transparent;
}
/ To customize the arrow tip at left /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
border-bottom: 12px solid transparent;
border-right: 12px solid #000;
border-top: 12px solid transparent;
}
/ To customize the arrow tip at right /
.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
border-bottom: 12px solid transparent;
border-left: 12px solid #000;
border-top: 12px solid transparent;
}
`
```

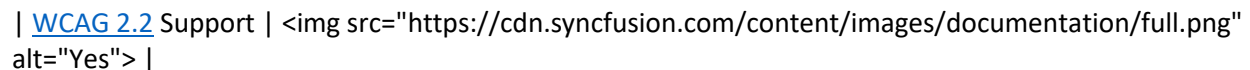
### Accessibility in ASP.NET MVC Tooltip component

The Tooltip component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Tooltip component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  alt="Yes"> |

| [Section 508](#) Support |  alt="Yes"> |

| Screen Reader Support |  alt="Yes"> |

```

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>

.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}

</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

### WAI-ARIA attributes

The Tooltip component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Tooltip component.

| Attributes       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| role="tooltip"   | The element that serves as the container for the tooltip has the ARIA role of <code>tooltip</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| aria-describedby | This attribute is added to the target element on which the Tooltip gets opened, when focusing or hovering over it. It usually holds the randomly generated Id value of the Tooltip element.<br>In case, the target element already holds an <code>aria-describedby</code> attribute with Id value of some other component, then the Tooltip Id value can be additionally appended to the existing <code>aria-describedby</code> attribute separated by a space as shown in the example below.<br><b>For example:</b> <code>aria-describedby = "my-text my-tooltip"</code> <code>my-text</code> is the Id of some other component. <code>my-tooltip</code> is the id of Tooltip component. When the Tooltip is closed, the <code>aria-describedby</code> attribute is removed from the target. |

| aria-hidden | This attribute is assigned to the Tooltip element whose default value is true. <br /><br /> When **true**, it denotes that the Tooltip element is in a hidden or a closed state. When the Tooltip appears on the screen, it's value changes to **false**. |

#### Keyboard interaction

The Tooltip component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Tooltip component.

| Keys | Description |

| --- | --- |

| Escape | Closes or dismisses the Tooltip. |

| Tab | A form control receiving focus (say through tab key), opens the Tooltip, and on focus out closes it. |

1. When the Tooltip is being displayed on the target element, focus continues to stay on it.
2. If the Tooltip opens on mouse entering into the target element space, then it should be dismissed only when the mouse leaves that target.
3. If the Tooltip opens on the target element that receives focus, then it should be closed only when the focus moves out of that target element.

Likewise, if the Tooltip opens on a click, then it should be closed only on another click action.

#### Ensuring accessibility

The Tooltip component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tooltip component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tooltip component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

#### Migration from Essential JS 1

This article describes the API migration process of Tooltip component from Essential JS 1 to Essential JS 2.

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Position | **Property:** *Position* <br/><br/> @Html.EJ().ToolTip("tooltip").Content("Tooltip content").Position(e => e.Target(target => target.Horizontal("right").Vertical("center")).Stem(e => e.Horizontal("left").Vertical("center"))) | **Property:** *Position* <br/><br/> @Html.EJS().ToolTip("tooltip").Position(Syncfusion.EJ2.Popups.Position.TopCenter) |

| Animation | **Property:** *Animation* <br/><br/> @Html.EJ().ToolTip("tooltip").Content("Tooltip content").Animation(e => e.Effect(Effect.Fade).Speed(200)) | **Property:** *Animation* <br/><br/> List<Object> animation = new List<Object>(); <br/> animation.Add(new { open = new { effect =

```
"FadeIn" }, close = new { effect = "fadeOut" } });
 ViewBag.animation = animation;

@Html.EJS().ToolTip("tooltip").Animation(ViewBag.animation) |
```

```
| Close Time Out | Property: AutoCloseTimeout


```

```
@Html.EJ().ToolTip("tooltip").Content("Tooltip
content").AutoCloseTimeout(2000).CloseMode(CloseMode.Auto) | Property: CloseDelay,
openDelay

 @Html.EJS().ToolTip("tooltip").CloseDelay(500) |
```

```
| Sticky Mode | Property: CloseMode

 @Html.EJ().ToolTip("tooltip").Content("Tooltip
content").CloseMode(CloseMode.Sticky) | Property: IsSticky

@Html.EJS().ToolTip("tooltip").IsSticky(true) |
```

```
| Offset from target | Not Applicable | Property: OffsetX/OffsetY

@Html.EJS().ToolTip("tooltip").OffsetX(10).OffsetY(10); |
```

```
| Open mode of tooltip | Not Applicable | Property: OpensOn

@Html.EJS().ToolTip("tooltip").OpensOn("Click") |
```

```
| Enable disable the tip of tooltip | Not Applicable | Property: ShowTipPointer

@Html.EJS().ToolTip("tooltip").ShowTipPointer(true) |
```

```
| Hide | Method: hide()

 @Html.EJ().ToolTip("tooltip").Content("Tooltip content")

 var tooltipObj = $("#tooltip").ejToolTip("instance");
 tooltipObj.hide(); |
```

```
Method: close()

 @Html.EJS().ToolTip("tooltip").OpensOn("Custom")

 var
tooltipObj = document.getElementById('tooltip').ej2_instances[0];
 tooltipObj.close(); |
```

```
| Show | Method: show()

 @Html.EJ().ToolTip("tooltip").Content("Tooltip content")

 var tooltipObj = $("#tooltip").ejToolTip("instance");
 tooltipObj.show(); |
```

```
Method: open()

 @Html.EJS().ToolTip("tooltip").OpensOn("Custom")

 var
tooltipObj = document.getElementById('tooltip').ej2_instances[0];
 tooltipObj.open(); |
```

```
| Enable | Method: enable()

 @Html.EJ().ToolTip("tooltip").Content("Tooltip content")

 var tooltipObj = $("#tooltip").ejToolTip("instance");
 tooltipObj.enable(); |
```

```
Method: destroy()

 @Html.EJS().ToolTip("tooltip").OpensOn("Custom")

var tooltipObj = document.getElementById('tooltip').ej2_instances[0];

tooltipObj.destroy(); |
```

```
| Close | Event: close

 @Html.EJ().ToolTip("tooltip").Content("Tooltip
content").ClientSideEvents(e => e.Close("onClose"))

 function onClose(args) { } |
```

```
Event: AfterClose

 @Html.EJS().ToolTip("tooltip").AfterClose("onAfterClose")

 function onAfterClose() { } |
```

```
| Open | Event: open

 @Html.EJ().ToolTip("tooltip").Content("Tooltip
content").ClientSideEvents(e => e.Open("onOpen"))

 function onOpen(args) { } |
```

```
Event: AfterOpen

 @Html.EJS().ToolTip("tooltip").AfterOpen("onAfterOpen")

 function onAfterOpen() { } |
```

```
| Before Collision | Not Applicable | Event: BeforeCollision


```

```
@Html.EJS().ToolTip("tooltip").BeforeCollision("onBeforeCollision")

 function
onBeforeCollision() { } |
```

```
| Tracking| Event: tracking

 @Html.EJ().ToolTip("sample").Content("Tooltip
content").ClientSideEvents(e => e.Tracking("onTracking")).Associate(Associate.MouseFollow)

 function onTracking(args) { } | Method: MouseTrail

@Html.EJS().ToolTip("tooltip").MouseTrail(true) |
```

## How To

### Show Tooltip on disabled elements and disable tooltip

By default, Tooltips will not be displayed on disabled elements. However, it is possible to enable this behavior by following the steps below.

1. Add a disabled element like the **button** element into a div whose display style is set to **inline-block**.
2. Set the pointer event as **none** for the disabled element (button) through CSS.
3. Now, initialize the Tooltip for outer div element that holds the disabled button element.

## CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div id="tooltip">
 @Html.EJS().ToolTip("box").Content("Tooltip from disabled
element").ContentTemplate(@<div>
 <div id="box" style="display: inline-block;">
 <input type="button" id="disabledbutton" disabled
value="Disabled button" />
 </div>
 </div>).Render()
</div>
<style>
 #tooltip {
 color: #008cff;
 height: 40px;
 left: 45%;
 position: absolute;
 top: 45%;
 width: 30%;
 }
 #disabledbutton {
 pointer-events: none;
 }
</style>
```

## DISABLED.CS

```
public ActionResult Disabled()
{
 ViewBag.content = "Tooltip from disabled element";
 return View();
}
```

### Dynamic Tooltip content with HTML elements

The Tooltip component loads HTML tags using the [content](#) template.



The HTML tags such as `<div>`, `<span>`, **bold**, *italic*, underline, etc., can be used. Style attributes can also be applied with HTML tags.

Here, Bold, Italic, Underline, and Anchor tags are used.

When using HTML elements as content to **ToolTip** make the content element `display: none`, then from the [beforeRender](#) event we can make the element visible again using below code.

```
`typescript
```

```
document.getElementById('content').style.display = 'block';
```

```
,
```

### CSHTML

```
@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@{
 var content = "<div id=\"tooltip\">" +
 "<h2>HTML Tags</h2>" +
 "Through templates, tooltip"
content can be loaded with <u><i> inline HTML, images, iframe,
videos, maps </i></u>. A title can be added to the content" +
 "</div>";
}
<div id="container">

@Html.EJS().ToolTip("target").CssClass("customtooltip").Content(content).ContentTemplate(@<div id="tooltipContent">
 <div class="content">
 <button class="text" id="Title">HTML(With Title)</button>
 </div>
</div>).Render()
</div>
<style>
 #tooltipContent table {
 margin: 0 auto;
 }
 #tooltipContent {
 display: inline-block;
 position: relative;
 left: 50%;
 transform: translateX(-50%);
 margin-top: 100px;
 }
 .text {
 text-transform: capitalize;
 width: 155px;
 }
 .header {
 font-family: "Arial Black", Gadget, sans-serif;
 font-size: 12px;
 padding-bottom: 2px;
 margin: 4px -7px 7px -7px;
 padding-right: 5px;
 padding-left: 6px;
 font-weight: bold;
 }
</style>
```

```

 height: 18px;
 border-bottom: 1px solid white;
 }
 .e-tooltip-css.e-tooltip-wrap .e-tip-content {
 padding: 0 10px 10px 10px;
 }
</style>

```

### HTML-CONTENT.CS

```

public ActionResult TooltipView()
{
 return View();
}

```

### Custom Tooltip with dynamic HTML

Tooltip loads HTML pages via HTML tags such as iframe, video, and map using the [content](#) property, which supports both string and HTML tags.

To load an `iframe` element in tooltip, set the required iframe in the `content` of tooltip while initializing the tooltip component. Refer to the following code.

`typescript

content: '<iframe src="https://www.syncfusion.com/products/essential-js2"></iframe>'

,

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
@{
 var content = "<iframe
src=\"https://www.syncfusion.com/products/essential-js2\"
id=\"tooltipFrame\"></iframe>";
}
<div id='container'>
 <div id="tooltipContent">
 <div class="content">
 @Html.EJS().Tooltip("target").CssClass("e-tooltip-
css").OpensOn("Click").Width("350").Height("250").Content(content).ContentTe
mplate(@<button class="text e-btn e-primary e-outline"
id="iframeContent">Embedded Iframe</button>).Render()
 </div>
 </div>
</div>
<style>
 #tooltipContent {
 position: relative;
 left: 50%;
 transform: translateX(-50%);
 margin: 65px 10px;
 }
 .content {
 display: inline-block;

```

```

 width: 49%;
 }
 #tooltipFrame {
 width: 332px;
 height: 233px;
 }
 .content button {
 position: relative;
 left: 50%;
 transform: translateX(-50%);
 }
</style>

```

### HTML-PAGE.CS

```

public ActionResult TooltipView()
{
 return View();
}

```

#### Tooltip open or display modes

The open mode property of tooltip can be defined on a target that is hovering, focusing, or clicking.

Tooltip component have the following types of open mode:

- Auto
- Hover
- Click
- Focus
- Custom

#### *Auto*

Tooltip appears when you hover over the target or when the target element receives the focus.

#### *Hover*

Tooltip appears when you hover over the target.

#### *Click*

Tooltip appears when you click a target element.

#### *Focus*

Tooltip appears when you focus (say through tab key) on a target element.

#### *Custom*

Tooltip is not triggered by any default action. So, bind your own events and use either open or close public methods.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div id='container'>
 <div id="showTooltip">
 <div id="first">

```

```

@Html.EJS().ToolTip("target01").OpensOn("Hover").Content("Tooltip from
hover").ContentTemplate(@<button class="blocks e-btn e-primary e-outline"
id="tooltiphover">Hover me! (Default)</button>).Render()

@Html.EJS().ToolTip("target02").OpensOn("Click").Content("Tooltip from
click").ContentTemplate(@<button class="blocks e-btn e-primary e-outline"
id="tooltipclick">Click me!</button>).Render()
</div>

<div id="second">
 @Html.EJS().ToolTip("target3").OpensOn("Click").Content("Click
close icon to close me").IsSticky(true).ContentTemplate(@<button
class="blocks e-btn e-primary e-outline" id="Mode">Sticky
Mode</button>).Render()
 @Html.EJS().ToolTip("target4").OpensOn("Click").Content("Opens
and closes Tooltip with delay of <i>1000
milliseconds</i>").OpenDelay(1000).CloseDelay(1000).ContentTemplate(@<button
class="blocks e-btn e-primary e-outline" id="openMode">Tooltip with
delay</button>).Render()
</div>

</div>
</div>
<style>
 #container {
 width: 100%;
 }
 #textbox {
 display: inline-block;
 top: -3px;
 }
 .blocks {
 margin: 0 10px 0 10px;
 text-transform: none;
 width: 168px;
 }
 #showTooltip {
 display: table;
 padding-top: 40px;
 margin: 0 auto;
 }
 #focus {
 border: 1px solid #ff4081;
 text-align: center;
 height: 31px;
 width: 168px;
 }
 ::placeholder {
 color: #ff4081;
 }
</style>

```

**OPEN-CUSTOM.CS**

```
public ActionResult TooltipView()
```

```
{
 return View();
}
```

### Fancy Tooltip Customization

The arrow of the tooltip can be customized as needed by customizing CSS in the sample-side. The EJ2 tooltip component is achieved through CSS3 format and positioned the tip arrow according to the tooltip positions like **TopCenter**, **BottomLeft**, **RightTop**, and more.

Here, the tip arrow is customized as Curved tooltip and Bubble tooltip.

#### Curved tip

The content for the tip pointer arrow has been added. To customize the curved tip arrow, override the following CSS class of tip arrow.

```
`typescript
cssClass = 'curvetips e-tooltip-css',
,

`css
.e-arrow-tip-outer.e-tip-bottom,
.e-arrow-tip-outer.e-tip-top {
border-left: none !important;
border-right: none !important;
border-top: none !important;
}
.e-arrow-tip.e-tip-top {
transform: rotate(170deg);
}
,
```

#### Bubble tip

The two **divs**(inner div and outer div) have been added to achieve the bubble tip arrow. To customize the bubble tip arrow, override the following CSS class of tip arrow.

```
`js
cssClass: 'bubbletip e-tooltip-css'
,

`css
.e-arrow-tip-outer.e-tip-bottom, .e-arrow-tip-outer.e-tip-top
{
border-radius: 50px;
```

```

height: 10px;
width: 10px;
}
.e-arrow-tip-inner.e-tip-bottom, .e-arrow-tip-inner.e-tip-top
{
border-radius: 50px;
height: 10px;
width: 10px;
}
`

```

These tip arrow customizations have been achieved through CSS changes in the sample level. The tooltip position can be changed by using the radio button click event.

The arrow tip pointer can also be disabled by using the [showTipPointer](#) property in a tooltip.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div id='container'>
 <div id="customization" class="customTipContainer">
 @Html.EJS().Tooltip("arrow").Content("Tooltip arrow
customized").CssClass("curvetips e-tooltip-css").ContentTemplate(@<button
id="target">
 Customized Tip Arrow
 </button>).Render()
 <div id="positions">

@Html.EJS().RadioButton("element1").Label("TopCenter").Value("TopCenter").Ch
ecked(true).Change("onChange").Render()

@Html.EJS().RadioButton("element2").Label("BottomLeft").Value("BottomLeft").
Checked(false).Change("onChange").Render()

 </div>
 </div>
 <div id="balloon" class="customTipContainer">
 @Html.EJS().Tooltip("bubble").Content("Tooltip arrow customized as
balloon tip").CssClass("bubbletip e-tooltip-css").ContentTemplate(@<button
id="bubbletip">
 Bubble Tip Arrow
 </button>).Render()
 <div id="btn">


```

```

@Html.EJS().RadioButton("radio1").Label("BottomLeft").Value("BottomLeft").Checked(false).Change("onChanged").Render()

@Html.EJS().RadioButton("radio2").Label("TopRight").Value("TopRight").Checked(true).Change("onChanged").Render()

</div>
</div>
<div id="disabledContainer" class="customTipContainer">
 @Html.EJS().ToolTip("tooltip").Content("Disabled tooltip pointer").CssClass("pointertip e-tooltip-css").ContentTemplate(@<button id="tooltip">
 Disabled Tip Arrow
 </button>).Render()
</div>
</div>
<script>
 function onChange(args) {
 document.getElementById('arrow').ej2_instances[0].position = args.value;
 document.getElementById('arrow').ej2_instances[0].dataBind();
 }
 function onChanged(args) {
 var bubble = document.getElementById('bubble').ej2_instances[0];
 bubble.position = args.value;
 if (bubble.position == 'BottomLeft') {
 bubble.offsetY = -30;
 } else {
 bubble.offsetY = 0;
 }
 bubble.dataBind();
 }
</script>
<style>
 #bubbleletip {
 border-color: #d2a679;
 }
 #target {
 border-color: #e86238;
 }
 li {
 list-style: none;
 }
 .e-radio-wrapper {
 margin-top: 18px;
 }
 #target,
 #bubbleletip,
 #tooltip {
 box-sizing: border-box;
 padding: 20px;
 width: 200px;
 text-align: center;
 }

```

```

 top: -17px;
 margin-bottom: 40px;
 }
 /* csslint ignore:start */
 @@font-face {
 font-family: "tip";
 src:
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.ttf") format("truetype"),
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.woff") format("woff"),
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.eot") format("eot"),
url("https://ej2.syncfusion.com/products/typescript/tooltip/customization/Fo
nts/tip.svg?#tip") format("svg");
 font-weight: normal;
 font-style: normal;
 }
 /* csslint ignore:end */
 /* csslint ignore:start */
 #container {
 width: 100%;
 }
 .customTipContainer {
 width: 400px;
 position: relative;
 left: 50%;
 transform: translateX(-50%);
 top: 50px;
 }
 #disabledContainer {
 margin-top: 25px;
 }
 .pointertip.e-tooltip-wrap .e-tip-content,
 .curvetips.e-tooltip-wrap .e-tip-content {
 color: white;
 }
 .pointertip.e-tooltip-wrap.e-popup {
 background-color: #80180d;
 border: 3px solid #ff9999;
 }
 .curvetips .e-arrow-tip.e-tip-top {
 margin-left: -20px;
 top: -16px;
 transform: rotate(177deg);
 left: 50px;
 }
 .curvetips.e-tooltip-wrap {
 padding: 17px;
 border-radius: 5px;
 }
 .curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom:before,
 .curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top:before {
 font-family: "tip" !important;
 speak-as: none;
 font-size: 21px;
 font-style: normal;
 }

```



```

 font-weight: normal;
 font-variant: normal;
 text-transform: none;
 line-height: 1;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
 content: "\e700";
 color: #e86238;
 }
 .curvetips.e-tooltip-wrap.e-popup {
 background: #e86238;
 border: none;
 }
 .curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom,
 .curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
 border-left: none;
 border-right: none;
 border-top: none;
 }
 .curvetips.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom:before,
 .curvetips.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top:before {
 content: none;
 }
 .curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom,
 .curvetips.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
 top: -1px;
 }
 .curvetips.e-tooltip-wrap .e-arrow-tip.e-tip-bottom,
 .curvetips.e-tooltip-wrap .e-arrow-tip.e-tip-top {
 position: absolute;
 height: 18px;
 width: 28px;
 }
 #positions {
 display: inline-block;
 }
 #btn {
 display: inline-block;
 }
 #target .e-tip-content {
 padding: 0px;
 }
 .bubbletip.e-tooltip-wrap {
 padding: 8px;
 }
 .bubbletip.e-tooltip-wrap .e-tip-content {
 border-radius: 50%;
 text-align: center;
 color: white;
 }
 .bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-bottom {
 height: 40px;
 width: 50px;
 }
 .bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-top {
 height: 40px;
 width: 40px;
 }

```

```
}
.bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-left {
 height: 12px;
 width: 20px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip.e-tip-right {
 height: 12px;
 width: 20px;
}
.bubbletip.e-tooltip-wrap.e-popup {
 border: 5px solid #dfccad;
 background-color: #7b5e32;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-bottom {
 height: 10px;
 width: 10px;
 border: 1px solid #dfccad;
 background-color: #7b5e32;
 border-radius: 50px;
 margin-top: 20px;
 margin-right: 20px;
}
.e-arrow-tip.e-tip-top {
 margin-left: 60px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-top {
 border: 1px solid #dfccad;
 border-radius: 50px;
 background-color: #7b5e32;
 width: 10px;
 height: 10px;
 margin-left: 20px;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-left {
 border-bottom: 6px solid transparent;
 border-right: 20px solid #dfccad;
 border-top: 6px solid transparent;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-outer.e-tip-right {
 border-bottom: 6px solid transparent;
 border-left: 20px solid #dfccad;
 border-top: 6px solid transparent;
}
.bubbletip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-bottom {
 margin-top: -2px;
 margin-left: 8px;
 content: none;
 top: 1px !important;
 border: 2px solid #dfccad;
 width: 20px;
 height: 20px;
 border-radius: 50px;
 background-color: #7b5e32;
}
.bubbletip .e-arrow-tip.e-tip-top {
 left: 44px !important;
 top: -19px !important;
```

```

 }
 .bubbletip .e-arrow-tip.e-tip-bottom {
 top: 88.9% !important;
 left: 4px !important;
 }
 .bubbletip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top {
 top: 10px !important;
 border: 2px solid #dfccad;
 width: 20px;
 height: 20px;
 border-radius: 50px;
 background-color: #7b5e32;
 }
 .bubbletip.e-tooltip-wrap .e-arrow-tip-inner.e-tip-top:before {
 content: None;
 }
 .bubbletip.e-tooltip-wrap .e-tip-content {
 border-radius: inherit;
 }
 .bubbletip.e-tooltip-wrap.bubbletip {
 width: 150px !important;
 border-radius: 50%;
 }
 /* csslint ignore:end */
</style>

```

### CUSTOM-TOOLTIP.CS

```

public ActionResult MouseTrailing()
{
 return View();
}

```

### Display Tooltip on SVG and canvas elements

Tooltip can be displayed on both SVG and Canvas elements. You can directly attach the `<svg>` or `<canvas>` elements to show tooltips on data visualization elements.

### CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div id="container">
 <div id="box" class="e-prevent-select">
 @Html.EJS().Tooltip("tooltip01").Content("SVG
Square").Target("#rectShape").ContentTemplate(@<div class="circletool"
id="rectShape" style="left:1%;top:10%">
 <svg>
 <rect id="square" class="shape" x="50" y="20" width="50"
height="50" style="fill:#FDA600;stroke:none;stroke-width:5;stroke-
opacity:0.9" />
 </svg>
 </div>).Render()
 @Html.EJS().Tooltip("tooltip02").Content("SVG
Ellipse").Target("#ellipseShape").ContentTemplate(@<div class="circletool"
id="ellipseShape" style="top:65%;">

```

```

 <svg>
 <ellipse id="ellipse" class="shape" cx="100" cy="50" rx="20"
ry="40" style="fill:#0450C2;stroke:none;stroke-width:2" />
 </svg>
 </div>).Render()
 @Html.EJS().ToolTip("tooltip03").Content("SVG
PolyShape").Target("#polyShape").ContentTemplate(@<div class="circletool"
id="polyShape" style="top:25%;left:40%">
 <svg>
 <polyline id="polyline" class="shape" points="0,40 40,40
40,80 80,80 80,120 120,120 120,160"
style="fill:#ffffff;stroke:#0450C2;stroke-width:4" />
 </svg>
 </div>).Render()
 @Html.EJS().ToolTip("tooltip04").Content("Canvas
Circle").Target("#circleShape").ContentTemplate(@<div class="circletool"
id="circleShape" style="top:16%;left:72%">
 <canvas id="circle" class="shape" width="60"
height="60"></canvas>
 </div>).Render()
 @Html.EJS().ToolTip("tooltip05").Content("Canvas
Triangle").Target("#triShape").ContentTemplate(@<div class="circletool"
id="triShape" style="top:73%;left:76%">
 <canvas id="triangle" class="shape" width="100"
height="50"></canvas>
 </div>).Render()
</div>
</div>
<script>
 window.onload = function () {
 var canvas = document.getElementById('triangle');
 var context;
 if (canvas.getContext) {
 context = canvas.getContext('2d');
 context.beginPath();
 context.moveTo(0, 50);
 context.lineTo(100, 50);
 context.lineTo(50, 0);
 context.fillStyle = "#FDA600";
 context.fill();
 }
 canvas = document.getElementById('circle');
 context = canvas.getContext('2d');
 var centerX = canvas.width / 2;
 var centerY = canvas.height / 2;
 var radius = 30;
 context.beginPath();
 context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
 context.fillStyle = '#0450C2';
 context.fill();
 };
</script>
<style>
 @@media (max-width: 500px) {
 #rectShape {
 /* csslint ignore:start */
 left: -11% !important;

```

```

 /* csslint ignore:end */
 }
 #ellipseShape {
 /* csslint ignore:start */
 left: -20% !important;
 /* csslint ignore:end */
 }
 #polyShape {
 /* csslint ignore:start */
 left: 28% !important;
 /* csslint ignore:end */
 }
 #circleShape {
 /* csslint ignore:start */
 left: 68% !important;
 /* csslint ignore:end */
 }
 #triShape {
 /* csslint ignore:start */
 left: 65% !important;
 /* csslint ignore:end */
 }
}
@media (min-width: 500px) and (max-width: 600px) {
 #triShape {
 /* csslint ignore:start */
 left: 70% !important;
 /* csslint ignore:end */
 }
}
#container {
 width: 80%;
 margin: 0 auto;
}
.e-tooltip-css {
 filter: drop-shadow(2px 5px 5px rgba(0, 0, 0, 0.25));
}
#control-container {
 /* csslint ignore:start */
 padding: 0 !important;
 /* csslint ignore:end */
}
#box {
 border: 1px solid #dddddd;
 background: #ffffff;
 box-sizing: border-box;
 height: 320px;
 position: relative;
}
.circletool {
 position: absolute;
}
</style>

```

```
public ActionResult MouseTrailing()
{
 return View();
}
```

### Dynamic Tooltip content

The tooltip content can be changed dynamically using the Fetch request.

The Fetch request should be made within the [beforeRender](#) event of the tooltip. On every success, the corresponding retrieved data will be set to the [content](#) property of the tooltip.

When you hover over the icons, its respective data will be retrieved dynamically and then assigned to the tooltip's content.

Refer to the following code snippet to change the tooltip content dynamically.

```
`js
function onBeforeRender(args): void {
 this.content = 'Loading...';
 this.dataBind();
 var fetchApi = new Fetch('./tooltip.json', 'GET');
 fetchApi.send().then(
 (result: any) => {
 for (var i: number = 0; i < result.length; i++) {
 if (result[i].Id == args.target.id) {
 / tslint:disable /
 this.content = result[i].Sports;
 / tslint:enable /
 }
 }
 this.dataBind();
 },
 (reason: any) => {
 this.content = reason.message;
 this.dataBind();
 });
 }
}
```

### CSHTML

```
@using Syncfusion.EJ2
```

```

@using Syncfusion.EJ2.Popups
<div id="container">
 <h2> Dynamic Tooltip content </h2>

@Html.EJS().Tooltip("Tooltip").BeforeRender("onBeforeRender").Position(Position.TopCenter).Target(".circletool").ContentTemplate(@<div id="box"
class="e-prevent-select">
 <div id="1" class="circletool bold-01" style="display:inline-block"></div>
 <div id="2" class="circletool italic" style="display:inline-block"></div>
 <div id="3" class="circletool underline-02" style="display:inline-block"></div>
 <div id="4" class="circletool cut-02" style="display:inline-block"></div>
 <div id="5" class="circletool copy" style="display:inline-block"></div>
 <div id="6" class="circletool paste" style="display:inline-block"></div>
</div>).Render()
</div>
<script>
 function onBeforeRender(args) {
 this.content = 'Loading...';
 this.dataBind();
 var fetchApi = new ej.base.Fetch('./tooltip.json', 'GET');
 fetchApi.send().then(
 (result) => {
 for (let i = 0; i < result.length; i++) {
 if (result[i].Id == args.target.id) {
 this.content = result[i].Name;
 }
 }
 this.dataBind();
 },
 (reason) => {
 this.content = reason.message;
 this.dataBind();
 }
);
 }
</script>
<style>
 h2 {
 text-align: center;
 margin-bottom: 50px;
 }
 #box {
 height: 80px;
 position: relative;
 text-align: center;
 }
 .circletool {
 height: 35px;
 width: 35px;
 }
 .underline-02 {

```

```

background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 48 48'%3E%3Cpath fill='#000'
d='M16.163 34.044H31.84V37H16.163zM27.02 11h7.594v1.349h-1.127c-.62 0-
1.058.24-1.393.755-.072.11-.233.484-.233 1.81v7.76c0 1.957-.197 3.507-.59
4.61-.414 1.137-1.22 2.123-2.405 2.928-1.175.802-2.79 1.204-4.78 1.204-2.16
0-3.84-.39-4.985-1.157a6.293 6.293 0 0 1-2.459-3.12c-.32-.892-.482-2.478-
.482-4.854v-7.483c0-1.404-.239-1.88-.377-2.043-.248-.273-.672-.411-1.267-
.411h-1.13V11h9.09v1.347h-1.144c-.653 0-1.094.184-1.354.569-.105.156-
.279.598-.279 1.885v8.344c0 .723.063 1.563.198 2.513.897.354 1.592.676
2.069.32.48.788.878 1.392 1.192.613.314 1.378.473 2.293.473 1.176 0 2.237-
.257 3.159-.767.899-.497 1.521-1.135 1.85-1.894.335-.798.503-2.2.503-4.167v-
7.751c0-1.522-.219-1.893-.29-1.972-.282-.325-.721-.481-1.329-.481h-
1.13V11z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.bold-01 {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 48 48'%3E%3Cpath fill='#000'
d='M18.637 25.625v6.987h3.9c1.95 0 3.088 0 3.576-.162.812-.162 1.462-.488
1.95-.975.487-.488.812-1.3.812-2.275 0-.812-.163-1.463-.488-1.95-.325-.487-
.974-.975-1.625-1.138-.974-.324-2.437-.487-4.875-.487zm0-
10.4V21.4h4.713c2.112 0 1.138 0 1.625-.163.975-.162 1.625-.487 2.112-
.974a2.691 2.691 0 0 0 .813-1.95c0-.813-.162-1.463-.65-1.95-.488-.488-1.3-
.813-2.113-.975-.487 0 .163-.163-2.274-.163zM13.762 11h9.1c2.113 0 3.575.163
4.55.325.975.162 1.95.488 2.763 1.137.813.488 1.462 1.3 1.95
2.113.488.975.812 1.95.812 3.088 0 1.137-.324 2.274-.975 3.412-.65.975-1.625
1.787-2.6 2.275 1.626.487 2.763 1.3 3.575 2.438.813 1.137 1.3 2.437 1.3 3.9
0 1.137-.324 2.274-.812 3.412-.488 1.137-1.3 1.95-2.275 2.6a7.813 7.813 0 0
1-3.575 1.3c-.813-.163-.813 0-4.063 0h-9.75z'/%3E%3C/svg%3E") no-repeat 100%
100%;
}
.italic {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 48 48'%3E%3Cpath fill='#000'
d='M22.434 11h10.504l-.398 1.333-.269-.003c-.804 0-1.3.078-1.615.228-
.513.232-.9.546-1.155.953-.28.431-.648 1.44-1.099 2.984l-4.417 15.298c-.484
1.71-.584 2.36-.584 2.606 0
.247.052.443.17.608.122.158.32.293.6.387.217.077.747.182
2.06.273l.404.027L26.249 37H15.063l.507-1.324.216-.009c1.248-.026 1.751-.142
1.96-.232.503-.197.865-.452 1.087-.772.391-.558.811-1.582 1.238-3.051.43-
15.298c.367-1.235.548-2.174.548-2.789 0-.26-.053-.465-.181-.635-.123-.17-
.327-.307-.59-.402-.204-.075-.688-.16-1.839-.16h-.437z'/%3E%3C/svg%3E") no-
repeat 100% 100%;
}
.cut-02 {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 48 48'%3E%3Cpath fill='#000'
d='M21.941 27.375c-1.393-.028-3.037 1.193-3.48 2.814-.325 1.138-.325
1.952.163 2.927.163.488.65 1.139 1.464 1.301.813.163 1.463.163 2.276-
.325.651-.488 1.301-1.3 1.627-2.276.325-1.139.325-1.952-.163-2.928-.325-
.487-.65-1.138-1.464-1.463a2.14 2.14 0 0 0-.423-.05zm-6.645-8.838a4.983
4.983 0 0 0-1.55.269c-.65.325-1.302.813-1.464 1.626-.163.813 0 1.464.162
1.951.651.814 1.301 1.464 2.44 1.79 2.114.65 3.903-.326 4.228-1.627.488-.976
0-1.626-.163-2.114-.65-.813-1.3-1.464-2.439-1.789a6.847 6.847 0 0 0-1.214-
.106zM31.47 12.31-5.367 9.431.164.489L37 22.546l-2.602 1.464-8.781.812-
1.952.976 1.301 2.603c.488.975.488 2.276.163 3.577-.813 2.602-3.252 4.228-
5.366 3.578-2.277-.488-3.415-3.09-2.603-5.692.326-1.463 1.302-2.602 2.44-
3.0912.44-1.3-.977-1.79-2.439 1.3c-1.138.49-2.44.652-3.903.327-2.602-.814-

```



```

4.228-2.928-3.577-5.204.65-2.277 3.415-3.415 6.016-2.765 1.302.325 2.44
1.301 3.09 2.4411.138 2.114 1.79-.976 5.691-7.318z'/%3E%3C/svg%3E") no-
repeat 100% 100%;
}
.copy {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 48 48'%3E%3Cpath fill='#000'
d='M26.612 20.392v3.361s-.064 2.208-2.504 2.016l-2.85.014v9.303a.38.38 0 0 0
.38.384h10.71c.211 0 .38-.17.38-.384h.003V20.772a.38.38 0 0 0-.38-.38zm-
4.53-7.537v4.079s-.075 2.678-3.033 2.447l-3.46.015V30.68c0
.255.21.463.47.463h3.666v-5.763l4.768-5.054h.013l1.397-1.467h3.603v-
5.544a.461.461 0 0 0-.463-.461zM21.23 11h7.813a2.321 2.321 0 0 1 2.323
2.318v5.545h.981.003-.002c1.053 0 1.911.856 1.911 1.912v14.313A1.914 1.914 0
0 1 32.35 37H21.638a1.914 1.914 0 0 1-1.912-1.914V33H16.06a2.32 2.32 0 0 1-
2.321-2.32V18.906l5.779-6.124h.023z'/%3E%3C/svg%3E") no-repeat 100% 100%;
}
.paste {
background: transparent url("data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' viewBox='0 0 48 48'%3E%3Cpath fill='#000'
d='M24.652 23.171v13.045l9.039-.016V25.67h-2.513v-2.499zm-.84-
.78h7.368l3.326 3.28v11.313L23.812 37zm-10.318-8.845h2.463c-.005.033-
.01.065-.01.097v1.217c0 .678.553 1.232 1.229 1.232h8.3c.68 0 1.23-.554 1.23-
1.232v-1.217c0-.032-.008-.062-.01-.097h2.462v8.362H23.2v9.322h-9.706zm7.063-
1.398v1.602h1.86v-1.602zM20.073 11h2.763c.37 0
.666.345.666.77v1.103h1.972a.77.77 0 0 1 .769.77v1.216c0 .425-.348.77-
.77.77h-8.297a.77.77 0 0 1-.769-.77v-1.217a.77.77 0 0 1 .769-
.77h2.228V11.77c0-.423.298-.769.67-.769z'/%3E%3C/svg%3E") no-repeat 100%
100%;
}
</style>

```

## DYNAMIC-CONTENT.CS

```

public ActionResult MouseTrailing()
{
 return View();
}

```

## Create and show Tooltip on multiple targets

Tooltip can be created and shown on multiple targets within a container by defining the specific target elements to the target property. So, the tooltip is initialized only on matched targets within a container.

## CSHTML

```

@using Syncfusion.EJ2
@using Syncfusion.EJ2.Popups
<div id="container">
 <form id="details" role="form">
 <div id="user">
 <div class="info">User Name:</div>
 <div class="inputs">

@Html.EJS().Tooltip("tooltip1").Position(Position.RightCenter).Target("#unam
e").ContentTemplate(@<input type="text" id="uname" class="e-info e-input"
name="firstname" title="Please enter your name" />).Render()

```

```

 </div>
 </div>

 <div>
 <div class="info">Email Address:</div>
 <div class="inputs">

@Html.EJS().ToolTip("tooltip2").Position(Position.RightCenter).Target("#mail
").ContentTemplate(@<input type="text" id="mail" class="e-info e-input"
name="email" title="Enter a valid email address" />).Render()
 </div>
 </div>

 <div>
 <div class="info">Password:</div>
 <div class="inputs">

@Html.EJS().ToolTip("tooltip3").Position(Position.RightCenter).Target("#pwd"
).ContentTemplate(@<input id="pwd" type="password" class="e-info e-input"
name="password" title="Be at least 8 characters length" />).Render()
 </div>
 </div>

 <div>
 <div class="info">Confirm Password:</div>
 <div class="inputs">

@Html.EJS().ToolTip("tooltip4").Position(Position.RightCenter).Target("#cpwd
").ContentTemplate(@<input id="cpwd" type="password" class="e-info e-input"
name="Cpwd" title="Re-enter your password">).Render()
 </div>
 </div>

 <div class="btn">
 <input id="sample" type="button" class="e-btn" value="Submit" />
 <input id="clear" type="reset" value="Reset" class="e-btn" />
 </div>
</form>
</div>
<script>
 document.getElementById('sample').addEventListener('click', function ()
 {
 var tooltip1 = document.getElementById('tooltip1').ej2_instances[0];
 var tooltip3 = document.getElementById('tooltip3').ej2_instances[0];
 var name = document.getElementById('uname').value;
 var pwd = document.getElementById('pwd').value;
 var cpwd = document.getElementById('cpwd').value;
 if (name.length > 0 & name.length < 4) {
 document.getElementById('uname').title = 'Required Minimum 4
Characters';
 document.getElementById('uname').style.backgroundColor = 'red';
 var target = document.getElementById('uname');
 tooltip1.open(target);
 } else {
 document.getElementById('uname').style.backgroundColor =
'white';
 }
 }
)

```

```

 if (pwd !== '' && pwd.length < 8) {
 document.getElementById('pwd').title = 'Required Minimum 8
Characters';
 document.getElementById('pwd').style.backgroundColor = 'red';
 var pwdtarget = document.getElementById('pwd');
 tooltip3.open(pwdtarget);
 } else {
 document.getElementById('pwd').style.backgroundColor = 'white';
 }
 if (name.length >= 4 && pwd !== '' && pwd.length >= 8 && pwd ==
cpwd) {
 alert('Form Submitted');
 } else {
 alert('Details are not Valid');
 }
 })
 document.getElementById('clear').addEventListener('click', function () {
 var tooltip1 = document.getElementById('tooltip1').ej2_instances[0];
 var tooltip3 = document.getElementById('tooltip3').ej2_instances[0];
 document.getElementById('uname').style.backgroundColor = 'white';
 document.getElementById('pwd').style.backgroundColor = 'white';
 var target = document.getElementById('uname');
 tooltip1.close(target);
 document.getElementById('uname').title = 'Please enter your name';
 var pwdtarget = document.getElementById('pwd');
 tooltip3.close(pwdtarget);
 });
</script>
<style>
 #details .info {
 font-weight: bold;
 width: 165px;
 display: inline-block;
 margin-left: 10px;
 }
 #details .inputs {
 display: inline-block;
 }
 #details .btn {
 margin-top: 10px;
 position: relative;
 left: 50%;
 transform: translateX(-50%);
 display: inline-block;
 }
 #details #sample {
 margin-left: 10px;
 }
 #details #clear {
 margin-left: 10px;
 }
 #details {
 padding-top: 30px;
 padding-bottom: 30px;
 position: relative;
 left: 50%;
 transform: translateX(-50%);

```

```
 display: inline-block;
 }
</style>
```

### **MULTI-TARGET.CS**

```
public ActionResult MouseTrailing()
{
 return View();
}
```

## Tree Grid

### Getting Started with ASP.NET MVC Tree Grid Control

This section briefly explains about how to include [ASP.NET MVC Tree Grid](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

### **PACKAGE MANAGER**

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

```
<namespaces>
<add namespace="Syncfusion.EJ2"/>
</namespaces>
```

### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

#### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

### Add ASP.NET MVC Tree Grid control

Now, add the Syncfusion ASP.NET MVC Tree Grid control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Render()
```

### Defining Row Data

To bind data for the TreeGrid component, you can assign a `IEnumerable` object to the [DataSource](#) property. The list data source can also be provided as an instance of the **DataManager**.

#### CSHTML

```
@model List<TreeGridSample.Controllers.TreeGridItems>
@Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)Model).ChildMapping("Children").TreeColumnIndex(1).Render()
```

#### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(TreeGridItems.GetTreeData());
 }
}

public class TreeGridItems
{
 public TreeGridItems() { }
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public int Duration { get; set; }
 public List<TreeGridItems> Children { get; set; }
 public static List<TreeGridItems> GetTreeData()
 {
 List<TreeGridItems> BusinessObjectCollection = new
List<TreeGridItems>();
 TreeGridItems Record1 = null;
 Record1 = new TreeGridItems()
 {
 TaskId = 1,
 TaskName = "Planning",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5,
 Children = new List<TreeGridItems>(),
 };
 TreeGridItems Child1 = new TreeGridItems()
 {
 TaskId = 2,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child2 = new TreeGridItems()
 {
 TaskId = 3,
 TaskName = "Plan budget",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child3 = new TreeGridItems()
 {
 TaskId = 4,
 TaskName = "Allocate resources",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 Record1.Children.Add(Child1);
 Record1.Children.Add(Child2);
 Record1.Children.Add(Child3);
 TreeGridItems Record2 = new TreeGridItems()
 {
 TaskId = 6,
 TaskName = "Design",
 StartDate = new DateTime(2021, 08, 25),
```

```
 Duration = 3,
 Children = new List<TreeGridItems>()
 };
 TreeGridItems Child5 = new TreeGridItems()
 {
 TaskId = 7,
 TaskName = "Software Specification",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child6 = new TreeGridItems()
 {
 TaskId = 8,
 TaskName = "Develop prototype",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child7 = new TreeGridItems()
 {
 TaskId = 9,
 TaskName = "Get approval from customer",
 StartDate = new DateTime(2024, 06, 27),
 Duration = 2
 };
 Record2.Children.Add(Child5);
 Record2.Children.Add(Child6);
 Record2.Children.Add(Child7);
 BusinessObjectCollection.Add(Record1);
 BusinessObjectCollection.Add(Record2);
 return BusinessObjectCollection;
}
```

**Note:** [ChildMapping](#) specifies the mapping property path for subtasks in dataSource.

<br/> [TreeColumnIndex](#) specifies the index of the column that needs to have the expander button.

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Tree Grid control will be rendered in the default web browser.

| TaskId | TaskName    | StartDate       | Duration |
|--------|-------------|-----------------|----------|
| 1      | ▼ Planning  | Tue Jun 07 2... | 5        |
| 2      | Plan tim... | Tue Jun 07 2... | 5        |
| 3      | Plan bu...  | Tue Jun 07 2... | 5        |
| 4      | Allocate... | Tue Jun 07 2... | 5        |
| 6      | ▼ Design    | Wed Aug 25 ...  | 3        |
| 7      | Softwar...  | Wed Aug 25 ...  | 3        |
| 8      | Develop...  | Wed Aug 25 ...  | 3        |
| 9      | Get app...  | Thu Jun 27 2... | 2        |

### Defining Columns

The columns are automatically generated when columns declaration is empty or undefined while initializing the treegrid.

The TreeGrid has an option to define columns using [Columns](#) property. In [Column](#) property you have properties to customize columns.

Let's check the properties used here:

- The [Field](#) property is to map with a property name an array of JavaScript objects.
- The [HeaderText](#) property is to change the title of columns.
- The [TextAlign](#) property is to change the alignment of columns. By default, columns will be left aligned. To change columns to right align, you need to define **textAlign** as **Right**.
- Using [Format](#) property, you can format number and date values to standard or custom formats. Here, you have defined it for the conversion of numeric values to currency.

### CSHTML

```
@model List<TreeGridSample.Controllers.TreeGridItems>
@Html.EJS().TreeGrid("Pager").DataSource((IEnumerable<object>)Model).Columns
(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(70).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(160).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(9
0).Add();
 col.Field("EndDate").HeaderText("End
Date").Width(90).Format("yMd").Add();
}
```



```
col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();
 col.Field("Progress").HeaderText("Progress").Width(80).Add();
 col.Field("Priority").HeaderText("Priority").Width(80).Add();
}).ChildMapping("Children").TreeColumnIndex(1).Render()
```

### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(TreeGridItems.GetTreeData());
 }
}
public class TreeGridItems
{
 public TreeGridItems() { }
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public int Duration { get; set; }
 public List<TreeGridItems> Children { get; set; }
 public static List<TreeGridItems> GetTreeData()
 {
 List<TreeGridItems> BusinessObjectCollection = new
List<TreeGridItems>();
 TreeGridItems Record1 = null;
 Record1 = new TreeGridItems()
 {
 TaskId = 1,
 TaskName = "Planning",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5,
 Children = new List<TreeGridItems>(),
 };
 TreeGridItems Child1 = new TreeGridItems()
 {
 TaskId = 2,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child2 = new TreeGridItems()
 {
 TaskId = 3,
 TaskName = "Plan budget",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child3 = new TreeGridItems()
 {
 TaskId = 4,
 TaskName = "Allocate resources",
 StartDate = new DateTime(2016, 06, 07),
```

```
 Duration = 5
 };
 Record1.Children.Add(Child1);
 Record1.Children.Add(Child2);
 Record1.Children.Add(Child3);
 TreeGridItems Record2 = new TreeGridItems()
 {
 TaskId = 6,
 TaskName = "Design",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3,
 Children = new List<TreeGridItems>()
 };
 TreeGridItems Child5 = new TreeGridItems()
 {
 TaskId = 7,
 TaskName = "Software Specification",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child6 = new TreeGridItems()
 {
 TaskId = 8,
 TaskName = "Develop prototype",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child7 = new TreeGridItems()
 {
 TaskId = 9,
 TaskName = "Get approval from customer",
 StartDate = new DateTime(2024, 06, 27),
 Duration = 2
 };
 Record2.Children.Add(Child5);
 Record2.Children.Add(Child6);
 Record2.Children.Add(Child7);
 BusinessObjectCollection.Add(Record1);
 BusinessObjectCollection.Add(Record2);
 return BusinessObjectCollection;
}
}
```

| Task ID | Task Name            | Start Date | End Date | Duration | Progress | Priority |
|---------|----------------------|------------|----------|----------|----------|----------|
| 1       | ▼ <b>Planning</b>    | 6/7/2016   |          | 5        |          |          |
| 2       | Plan timeline        | 6/7/2016   |          | 5        |          |          |
| 3       | Plan budget          | 6/7/2016   |          | 5        |          |          |
| 4       | Allocate resources   | 6/7/2016   |          | 5        |          |          |
| 6       | ▼ <b>Design</b>      | 8/25/2021  |          | 3        |          |          |
| 7       | Software Specific... | 8/25/2021  |          | 3        |          |          |
| 8       | Develop prototy...   | 8/25/2021  |          | 3        |          |          |
| 9       | Get approval fro...  | 6/27/2024  |          | 2        |          |          |

### Enable Paging

The paging feature enables users to view the treegrid record in a paged view. It can be enabled by setting the [AllowPaging](#) property to true. Pager can be customized using [PageSettings](#) property.

In root-level paging mode, paging is based on the root-level rows only, i.e., it ignores the child row count and it can be enabled by using the [PageSizeMode](#) property of [PageSettings](#).

### CSHTML

```
@model List<TreeGridSample.Controllers.TreeGridItems>
@Html.EJS().TreeGrid("Pager").AllowPaging().DataSource((IEnumerable<object>)
Model).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(70).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(160).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(9
0).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();
}).ChildMapping("Children").TreeColumnIndex(1).PageSettings(page =>
page.PageSizes(true).PageCount(2)).Render()
```

### HOMECONTROLLER.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(TreeGridItems.GetTreeData());
 }
}
```

```

}
public class TreeGridItems
{
 public TreeGridItems() { }
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public int Duration { get; set; }
 public List<TreeGridItems> Children { get; set; }
 public static List<TreeGridItems> GetTreeData()
 {
 List<TreeGridItems> BusinessObjectCollection = new
List<TreeGridItems>();
 TreeGridItems Record1 = null;
 Record1 = new TreeGridItems()
 {
 TaskId = 1,
 TaskName = "Planning",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5,
 Children = new List<TreeGridItems>(),
 };
 TreeGridItems Child1 = new TreeGridItems()
 {
 TaskId = 2,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child2 = new TreeGridItems()
 {
 TaskId = 3,
 TaskName = "Plan budget",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child3 = new TreeGridItems()
 {
 TaskId = 4,
 TaskName = "Allocate resources",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 Record1.Children.Add(Child1);
 Record1.Children.Add(Child2);
 Record1.Children.Add(Child3);
 TreeGridItems Record2 = new TreeGridItems()
 {
 TaskId = 6,
 TaskName = "Design",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3,
 Children = new List<TreeGridItems>()
 };
 TreeGridItems Child5 = new TreeGridItems()
 {
 TaskId = 7,

```

```
 TaskName = "Software Specification",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child6 = new TreeGridItems()
 {
 TaskId = 8,
 TaskName = "Develop prototype",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child7 = new TreeGridItems()
 {
 TaskId = 9,
 TaskName = "Get approval from customer",
 StartDate = new DateTime(2024, 06, 27),
 Duration = 2
 };
 Record2.Children.Add(Child5);
 Record2.Children.Add(Child6);
 Record2.Children.Add(Child7);
 BusinessObjectCollection.Add(Record1);
 BusinessObjectCollection.Add(Record2);
 return BusinessObjectCollection;
}
```

| Task ID                                                                                                        | Task Name                  | Start Date | Duration |
|----------------------------------------------------------------------------------------------------------------|----------------------------|------------|----------|
| 1                                                                                                              | ▼ Planning                 | 6/7/2016   | 5        |
| 2                                                                                                              | Plan timeline              | 6/7/2016   | 5        |
| 3                                                                                                              | Plan budget                | 6/7/2016   | 5        |
| 4                                                                                                              | Allocate resources         | 6/7/2016   | 5        |
| 6                                                                                                              | ▼ Design                   | 8/25/2021  | 3        |
| 7                                                                                                              | Software Specification     | 8/25/2021  | 3        |
| 8                                                                                                              | Develop prototype          | 8/25/2021  | 3        |
| 9                                                                                                              | Get approval from customer | 6/27/2024  | 2        |
| <div><div>⏪ &lt; 1 &gt; ⏩</div><div>10 ▼</div><div>Items per page</div><div>1 of 1 pages (8 items)</div></div> |                            |            |          |

### Enable Sorting

The sorting feature enables you to order the records. It can be enabled by setting the [AllowSorting](#) property as true. Sorting feature can be customized using [SortSettings](#) property.

**CSHTML**

```
@model List<TreeGridSample.Controllers.TreeGridItems>
@Html.EJS().TreeGrid("Pager").AllowPaging().AllowSorting().DataSource((IEnumerable<object>)Model).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(70).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(160).Add();
 col.Field("StartDate").HeaderText("Start Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(90).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}).ChildMapping("Children").TreeColumnIndex(1).PageSettings(page =>
page.PageSizes(true).PageCount(2)).Render()
```

**HOMECONTROLLER.CS**

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(TreeGridItems.GetTreeData());
 }
}

public class TreeGridItems
{
 public TreeGridItems() { }
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public int Duration { get; set; }
 public List<TreeGridItems> Children { get; set; }
 public static List<TreeGridItems> GetTreeData()
 {
 List<TreeGridItems> BusinessObjectCollection = new List<TreeGridItems>();
 TreeGridItems Record1 = null;
 Record1 = new TreeGridItems()
 {
 TaskId = 1,
 TaskName = "Planning",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5,
 Children = new List<TreeGridItems>(),
 };
 TreeGridItems Child1 = new TreeGridItems()
 {
 TaskId = 2,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child2 = new TreeGridItems()
```

```
{
 TaskId = 3,
 TaskName = "Plan budget",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
};
TreeGridItems Child3 = new TreeGridItems()
{
 TaskId = 4,
 TaskName = "Allocate resources",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
};
Record1.Children.Add(Child1);
Record1.Children.Add(Child2);
Record1.Children.Add(Child3);
TreeGridItems Record2 = new TreeGridItems()
{
 TaskId = 6,
 TaskName = "Design",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3,
 Children = new List<TreeGridItems>()
};
TreeGridItems Child5 = new TreeGridItems()
{
 TaskId = 7,
 TaskName = "Software Specification",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
};
TreeGridItems Child6 = new TreeGridItems()
{
 TaskId = 8,
 TaskName = "Develop prototype",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
};
TreeGridItems Child7 = new TreeGridItems()
{
 TaskId = 9,
 TaskName = "Get approval from customer",
 StartDate = new DateTime(2024, 06, 27),
 Duration = 2
};
Record2.Children.Add(Child5);
Record2.Children.Add(Child6);
Record2.Children.Add(Child7);
BusinessObjectCollection.Add(Record1);
BusinessObjectCollection.Add(Record2);
return BusinessObjectCollection;
}
```

| Task ID                      | Task Name                  | ↑ | Start Date | Duration |
|------------------------------|----------------------------|---|------------|----------|
| 1                            | ▼ <b>Planning</b>          |   | 6/7/2016   | 5        |
| 2                            | Plan timeline              |   | 6/7/2016   | 5        |
| 3                            | Plan budget                |   | 6/7/2016   | 5        |
| 4                            | Allocate resources         |   | 6/7/2016   | 5        |
| 6                            | ▼ <b>Design</b>            |   | 8/25/2021  | 3        |
| 7                            | Software Specification     |   | 8/25/2021  | 3        |
| 8                            | Develop prototype          |   | 8/25/2021  | 3        |
| 9                            | Get approval from customer |   | 6/27/2024  | 2        |
| ⏪   ⏩   1 of 1 pages   ⏪   ⏩ |                            |   |            |          |

### Enable Filtering

The filtering feature enables you to view reduced amount of records based on filter criteria. It can be enabled by setting the [AllowFiltering](#) property as true. Filtering feature can be customized using [FilterSettings](#) property.

By default, filtered records are shown along with its parent records. This behavior can be changed by using the [HierarchyMode](#) property of [FilterSettings](#).

### CSHTML

```
@model List<TreeGridSample.Controllers.TreeGridItems>
@Html.EJS().TreeGrid("Pager").AllowPaging().AllowSorting().AllowFiltering().
DataSource((IEnumerable<object>)Model).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(70).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(160).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(90).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();
}).ChildMapping("Children").TreeColumnIndex(1).PageSettings(page =>
page.PageSizes(true).PageCount(2)).Render()
```

### HOMECONTROLLER.CS

```
public class HomeController : Controller
```



```
{
 public ActionResult Index()
 {
 return View(TreeGridItems.GetTreeData());
 }
}

public class TreeGridItems
{
 public TreeGridItems() { }
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public int Duration { get; set; }
 public List<TreeGridItems> Children { get; set; }
 public static List<TreeGridItems> GetTreeData()
 {
 List<TreeGridItems> BusinessObjectCollection = new
List<TreeGridItems>();
 TreeGridItems Record1 = null;
 Record1 = new TreeGridItems()
 {
 TaskId = 1,
 TaskName = "Planning",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5,
 Children = new List<TreeGridItems>(),
 };
 TreeGridItems Child1 = new TreeGridItems()
 {
 TaskId = 2,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child2 = new TreeGridItems()
 {
 TaskId = 3,
 TaskName = "Plan budget",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child3 = new TreeGridItems()
 {
 TaskId = 4,
 TaskName = "Allocate resources",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 Record1.Children.Add(Child1);
 Record1.Children.Add(Child2);
 Record1.Children.Add(Child3);
 TreeGridItems Record2 = new TreeGridItems()
 {
 TaskId = 6,
 TaskName = "Design",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3,
```

```

 Children = new List<TreeGridItems>()
 };
 TreeGridItems Child5 = new TreeGridItems()
 {
 TaskId = 7,
 TaskName = "Software Specification",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child6 = new TreeGridItems()
 {
 TaskId = 8,
 TaskName = "Develop prototype",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
 };
 TreeGridItems Child7 = new TreeGridItems()
 {
 TaskId = 9,
 TaskName = "Get approval from customer",
 StartDate = new DateTime(2024, 06, 27),
 Duration = 2
 };
 Record2.Children.Add(Child5);
 Record2.Children.Add(Child6);
 Record2.Children.Add(Child7);
 BusinessObjectCollection.Add(Record1);
 BusinessObjectCollection.Add(Record2);
 return BusinessObjectCollection;
}

```

| Task ID                                                                                                                                                              | Task Name     | Start Date | Duration |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------------|----------|
|                                                                                                                                                                      | plan          |            |          |
| 1                                                                                                                                                                    | ▼ Planning    | 6/7/2016   | 5        |
| 2                                                                                                                                                                    | Plan timeline | 6/7/2016   | 5        |
| 3                                                                                                                                                                    | Plan budget   | 6/7/2016   | 5        |
| <div> <div> <div>⏪</div> <div>⏩</div> <div>1</div> <div>⏪</div> <div>⏩</div> </div> <div>12</div> <div>Items per page</div> <div>1 of 1 pages (3 items)</div> </div> |               |            |          |
| Task Name: plan                                                                                                                                                      |               |            |          |

**Note:** [View Sample in GitHub.](#)

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

## Data binding in ASP.NET MVC Tree Grid Component

The TreeGrid uses **DataManager**, which supports both RESTful JSON data services binding and local JavaScript object array binding. The [DataSource](#) property can be assigned either with the instance of **DataManager** or JavaScript object array collection.

It supports two kinds of data binding method:

- Local data
- Remote data

### Binding with ajax

You can use TreeGrid [DataSource](#) property to bind the data source to TreeGrid from external Fetch request. In the below code we have fetched the data source from the server with the help of Fetch request and provided that to [DataSource](#) property by using **onSuccess** event of the Fetch.

#### CSHTML

```
@Html.EJS().Button("button").Content("Bind Data").Render()
@Html.EJS().TreeGrid("Ajax").Columns(col =>
{
 col.Field("TaskID").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(90).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
}) .IdMapping("TaskID").ParentIdMapping("ParentItem").AllowPaging().PageSettings(p => p.PageSize(7)).TreeColumnIndex(1).Render()
<script>
 document.getElementById("button").addEventListener('click', () => {
 var treegrid = document.getElementById("Ajax").ej2_instances[0];
 treegrid.element.parentNode.insertBefore(button, treegrid.element);
 var fetch = new
ej.base.Fetch("https://ej2services.syncfusion.com/production/web-services/api/SelfReferenceData", "GET");
 treegrid.showSpinner();
 fetch.send();
 fetch.onSuccess = function (data) {
 treegrid.hideSpinner();
 treegrid.dataSource = data;
 };
 });
</script>
```

#### AJAXBIND.CS

```
public ActionResult AjaxBind()
{
 return View();
}
```

**Note:** If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

<br/> You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

If you bind the dataSource from this way, then it acts like a local dataSource. So you cannot perform any server side crud actions.

### Handling expandStateMapping

To denote the expand status of parent row, define the [ExpandStateMapping](#) property of tree grid.

The [ExpandStateMapping](#) property maps the field name in data source, that denotes whether parent record is in expanded or collapsed state and this is useful to render parent row in expanded or collapsed state based on this mapping property value in data source.

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").DataSource(dataManager => {
 dataManager.Url("/Home/UrlDatasource").Adaptor("UrlAdaptor")}
 .Columns(col =>
 {
 col.Field("TaskID").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(190).Add();
 col.Field("Duration").HeaderText("Duration").Width(90).Add();
 })
 .Height(400).HasChildMapping("isParent").IdMapping("TaskID").ParentIdMapping("ParentValue")
 TreeColumnIndex(1).ExpandStateMapping("IsExpanded").Render()
```

### EXPANDSTATEMAPPING.CS

```
public ActionResult ExpandStateMapping()
{
 return View();
}

public class TreeData
{
 public static List<TreeData> tree = new List<TreeData>();
 [System.ComponentModel.DataAnnotations.Key]
 public int TaskID { get; set; }
 public string TaskName { get; set; }
 public int Duration { get; set; }
 public int? ParentValue { get; set; }
 public bool? isParent { get; set; }
 public bool IsExpanded { get; set; }
 public TreeData() { }
 public static List<TreeData> GetTree()
 {
 if (tree.Count == 0)
 {
 int root = 0;
 for (var t = 1; t <= 500; t++)
 {
```

```

 Random ran = new Random();
 string math = (ran.Next() % 3) == 0 ? "High" : (ran.Next() %
2) == 0 ? "Release Breaker" : "Critical";
 string progr = (ran.Next() % 3) == 0 ? "Started" :
(ran.Next() % 2) == 0 ? "Open" : "In Progress";
 root++;
 int rootItem = root;
 tree.Add(new TreeData() { TaskID = rootItem, TaskName =
"Parent task " + rootItem.ToString(), isParent = true, IsExpanded = false,
ParentValue = null, Duration = ran.Next(1, 50) });
 int parent = root;
 for (var d = 0; d < 1; d++)
 {
 root++;
 string value = ((parent + 1) % 3 == 0) ? "Low" :
"Critical";

 int par = parent + 1;
 progr = (ran.Next() % 3) == 0 ? "In Progress" :
(ran.Next() % 2) == 0 ? "Open" : "Validated";
 int iD = root;
 tree.Add(new TreeData() { TaskID = iD, TaskName = "Child
task " + iD.ToString(), isParent = true, IsExpanded = false, ParentValue =
rootItem, Duration = ran.Next(1, 50) });
 int subparent = root;
 for (var c = 0; c < 500; c++)
 {
 root++;
 string val = ((subparent + c + 1) % 3 == 0) ? "Low"
: "Critical";

 int subchild = subparent + c + 1;
 string progress = (ran.Next() % 3) == 0 ? "In
Progress" : (ran.Next() % 2) == 0 ? "Open" : "Validated";
 int childID = root ;
 tree.Add(new TreeData() { TaskID = childID, TaskName
= "sub Child task " + childID.ToString(), isParent = false, IsExpanded =
false, ParentValue = subparent, Duration = ran.Next(1, 50) });
 }
 }
 }
 return tree;
}
}

```

You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Immutable Mode

The immutable mode optimizes the Tree Grid re-rendering performance by using the object reference and [deep compare](#) concept. When performing the Tree Grid actions, it will only re-render the modified or newly added rows and prevent the re-rendering of the unchanged rows.

To enable this feature, you have to set the `EnableImmutableMode` property as **true**.

**Note:** \* This feature uses the primary key value for data comparison. So, you need to provide the [IsPrimaryKey](#) column.

### CSHTML

```
@Html.EJS().TreeGrid("Pager").AllowPaging().DataSource((IEnumerable<object>)
ViewBag.DataSource).EnableImmutableMode(true).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(70).IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(160).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(90).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();
}).ChildMapping("Children").TreeColumnIndex(1).PageSettings(page =>
page.PageSizes(true).PageCount(2)).Render()
```

### IMMUTABLE.CS

```
public ActionResult Index()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

### Limitations

The following features are not supported in the immutable mode:

- Frozen rows and columns
- Row Template
- Detail Template
- Column reorder
- Virtualization

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Columns in ASP.NET MVC Tree Grid Component

The column definitions are used as the [DataSource](#) schema in the TreeGrid. This plays a vital role in rendering column values in the required format.

The treegrid operations such as sorting, filtering and searching etc. are performed based on column definitions. The [Field](#) property of the [Columns](#)

is necessary to map the data source values in TreeGrid Columns.

**Note:** 1. If the column [Field](#) is not specified in the dataSource, the column values will be empty.

2. If the [Field](#) name contains “dot” operator, it is considered as complex binding.

[TreeColumnIndex](#) property denotes the column that is used to expand and collapse child rows.

### Format

To format cell values based on specific culture, use the [Format](#) property of [Column](#). The TreeGrid uses [Internalization](#) library to format [number](#) and [date](#)

values.

### CSHTML

```
@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("Formatting").AllowPaging()
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("ID").HeaderText("Order ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("Name").HeaderText("Order Name").Width(180).Add();

 col.Field("Price").HeaderText("Price").Format("yMd").TextAlign(TextAlign.Right).Width(90).Format("c2").Add();
 })
 .Height(315).ChildMapping("Children").TreeColumnIndex(1).Render()
)
```

### FORMATTING.CS

```
public ActionResult Formatting()
{
 var treeData = ShipmentData.GetShipmentData();
 ViewBag.datasource = treeData;
 return View();
}
```

**Note:** By default, the [number](#) and [date](#) values are formatted in [en-US](#) locale.

### Number formatting

The number or integer values can be formatted using the below format strings.

Format | Description | Remarks

{ type:'date', format:'dd/MM/yyyy' } | 04/07/1996

{ type:'date', format:'dd.MM.yyyy' } | 04.07.1996

{ type:'date', skeleton:'short' } | 7/4/96

{ type:'dateTime', format:'dd/MM/yyyy hh:mm a' } | 04/07/1996 12:00 AM

{ type:'dateTime', format:'MM/dd/yyyy hh:mm:ss a' } | 07/04/1996 12:00:00 AM

### CSHTML

```
@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("DateFormatting")
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .Columns(col =>
```

```

 {
 col.Field("ID").HeaderText("Order
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("Name").HeaderText("Order Name").Width(180).Add();
 col.Field("OrderDate").HeaderText("Order
Date").Format("yMd").TextAlign(TextAlign.Right).Width(90).Add();

col.Field("Price").HeaderText("Price").TextAlign(TextAlign.Right).Width(80).
Format("C2").Add();
 }) .Height(315).ChildMapping("Children").TreeColumnIndex(1).Render()
)

```

### DATEFORMATTING.CS

```

public ActionResult Dateformatting()
{
 var treeData = ShipmentData.GetShipmentData();
 ViewBag.datasource = treeData;
 return View();
}

```

### Lock columns

You can lock columns by using [LockColumn](#) property of [Column](#). The locked columns will be moved to the first position. Also you can't reorder its position.

In the below example, Duration column is locked and its reordering functionality is disabled.

### CSHTML

```

@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("Lock")
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(TextAlign.Right).Width(90).Add();

col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(TextAlign.R
ight).LockColumn(true).Add();

 }) .Height(315).ChildMapping("Children").AllowReordering().TreeColumnIndex(2)
 .Render()
)

```

### LOCK.CS

```

public ActionResult Lock()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}

```



### Column type

Column type can be specified using the [Type](#) property in [Column](#). It specifies the type of data the column binds.

If the [Format](#) is defined for a column, the column uses [Type](#) to select the appropriate format option ([number](#) or [date](#)).

TreeGrid column supports the following types:

- string
- number
- boolean
- date
- datetime

**Note:** If the [Type](#) is not defined, it will be determined from the first record of the [DataSource](#).

### Checkbox column

To render checkboxes in existing column, you need to set [ShowCheckbox](#) property as **true**.

It is also possible to select the rows hierarchically using checkboxes in TreeGrid by enabling [AutoCheckHierarchy](#) property. When we check on any parent record checkbox then the child record checkboxes will get checked.

### CSHTML

```
@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("CheckboxSelection")
 .Height(400)
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .AutoCheckHierarchy(true)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").ShowCheckbox(true).Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(190).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render()
)
```

### CHECKBOX.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
}
```

```

 ViewBag.datasource = tree;
 return View();
}

```

### Controlling Tree Grid actions

You can enable or disable treegrid action for a particular column by setting the [AllowFiltering](#), and [AllowSorting](#) properties in [Column](#).

### CSHTML

```

@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("GridAction").AllowFiltering().AllowSorting()
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).AllowSorting(false).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").AllowFiltering(false).TextAlign(TextAlign.Right).Width(
90).Add();
 col.Field("Duration").HeaderText("Duration").Width(80).Add();
 }).Height(270).ChildMapping("Children").TreeColumnIndex(1).Render()
)

```

### GRIDACTION.CS

```

public ActionResult GridAction()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}

```

### Show/hide columns by external button

You can show or hide treegrid columns dynamically using external buttons by invoking the [showColumns](#) or [hideColumns](#) method.

### CSHTML

```

@using Syncfusion.EJ2.Grids
@Html.EJS().Button("button1").Content("SHOW").Render()
@Html.EJS().Button("button2").Content("HIDE").Render()
@ (Html.EJS().TreeGrid("ShowHide").DataSource((IEnumerable<object>)ViewBag.da
tasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(TextAlign.Right).Width(120).Add();
 }
)

```

```
col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(TextAlign.Right).Add();

}).Height(270).ChildMapping("Children").TreeColumnIndex(1).Render()
)
<script>
 document.getElementById("button1").addEventListener('click', () => {
 var treegrid = document.getElementById("ShowHide").ej2_instances[0];
 treegrid.showColumns(['Task ID', 'Duration']); //show by HeaderText
 });
 document.getElementById("button2").addEventListener('click', () => {
 var treegrid = document.getElementById("ShowHide").ej2_instances[0];
 treegrid.hideColumns(['Task ID', 'Duration']); //hide by HeaderText
 });
</script>
```

### SHOWHIDE.CS

```
public ActionResult ShowHide()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

### ValueAccessor

The [ValueAccessor](#) is used to access/manipulate the value of display data. You can achieve custom value formatting by using the [ValueAccessor](#).

### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("Value").DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("orderId").HeaderText("Order ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(180).ValueAccessor("orderFormatter").Add();
 col.Field("orderDate").HeaderText("Order Date").Format("yMd").TextAlign(TextAlign.Right).Width(90).Add();

 col.Field("price").HeaderText("Price").ValueAccessor("currencyFormatter").TextAlign(TextAlign.Right).Width(80).Add();
 }).Height(315).ChildMapping("subTasks").TreeColumnIndex(1).Render()
)
<script>
 function currencyFormatter(field, data, column) {
 return '€' + data['price'];
 }
 function orderFormatter(field, data, column) {
 return data[field] + '-' + data['category'];
 }
}
```

```
</script>
```

### VALUE.CS

```
public ActionResult Value()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

#### Display array type columns

You can bind an array of objects in a column by using the [ValueAccessor](#) property.

In this example, the name field has an array of two objects, FirstName and LastName. These two objects are joined and bound to a column using the

[ValueAccessor](#).

### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("ValueArray").DataSource((IEnumerable<object>)ViewBag.
datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();

 col.Field("name").HeaderText("Assignee").Width(90).ValueAccessor("orderForma
tter").TextAlign(TextAlign.Right).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(TextAlign.R
ight).Add();
 }).Height(315).ChildMapping("Children").TreeColumnIndex(1).Render()
)
<script>
 function orderFormatter(field, data, column) {
 return data[field].map(function (s) {
 return s.lastName || s.firstName
 }).join(' ');
 }
</script>
```

### VALUEARRAY.CS

```
public ActionResult ValueArray()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

### Expression column

You can achieve the expression column by using the [ValueAccessor](#) property.

#### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("Expression").DataSource((IEnumerable<object>)ViewBag.
datasource)
 .Columns(col =>
 {
 col.Field("orderId").HeaderText("Order
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(180).Add();

col.Field("units").HeaderText("Units").TextAlign(TextAlign.Right).Width(90).
Add();
 col.Field("unitPrice").HeaderText("Unit
Price").TextAlign(TextAlign.Right).Width(80).Format("c2").Add();

col.Field("price").HeaderText("Price").TextAlign(TextAlign.Right).ValueAcces
sor("totalPrice").Width(80).Format("c2").Add();
 }).Height(315).ChildMapping("subTasks").TreeColumnIndex(1).Render()
)
<script>
 function totalPrice(field, data, column) {
 return data.units * data.unitPrice;
 };
</script>
```

#### EXPRESSION.CS

```
public ActionResult Expression()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

### How to render boolean values as checkbox

To render boolean values as checkbox in columns, you need to set [DisplayAsCheckBox](#) property as **true**.

#### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("DisplayAsCheckBox").DataSource((IEnumerable<object>)V
iewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();

col.Field("Approved").HeaderText("Approved").DisplayAsCheckBox(true).Width("
90").Add();
 col.Field("Duration").HeaderText("Duration").Width(80).Add();
 }).Height(315).ChildMapping("Children").TreeColumnIndex(1).Render()
)
```

```
)
```

### DISPLAYASCHECKBOX.CS

```
public ActionResult DisplayAsCheckbox()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Rows in ASP.NET MVC Tree Grid Component

The row represents record details fetched from data source.

#### Customize rows

You can customize the appearance of a row by using the [RowDataBound](#) event.

The [RowDataBound](#) event triggers for every row. In the event handler, you can get the **args** which contains details of the row.

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").RowDataBound("rowDataBound").DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").TreeColumnIndex(1).AllowPaging().Render()
<script>
 function rowDataBound(args) {
 if (args.data['Duration'] == 5) {
 args.row.style.background = '#336c12';
 } else if (args.data['Duration'] > 6) {
 args.row.style.background = '#7b2b1d';
 }
 }
</script>
```

### CUSTOMIZEROWS.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
}
```

```
return View();
}
```

### Styling alternate rows

You can change the treegrid's alternative rows' background color by overriding the **.e-altrow** class.

```
`css
.e-treegrid .e-altrow {
background-color: #fafafa;
}
`
```

Refer to the following example.

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(275).EnableHover(false).DataSource((
IEnumerable<object>) ViewBag.datasource).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging(true).TreeColumnIndex(1).Render()
<style>
 .e-treegrid .e-altrow {
 background-color: #fafafa;
 }
</style>
```

### ALTERNATEROWS.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** Refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to learn how to present and manipulate data.

### Cell in ASP.NET MVC Tree Grid Component

#### Customize cell styles

The appearance of cells can be customized by using the [QueryCellInfo](#) event.

The [QueryCellInfo](#) event triggers for every cell. In that event handler, you can get **QueryCellInfoEventArgs** that contains the details of the cell.

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(300).DataSource((IEnumerable<object>)
)ViewBag.datasource).QueryCellInfo("customizeCell").Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(90).Add();
}).ChildMapping("Children").TreeColumnIndex(1).AllowPaging().Render()
<script>
 function customizeCell(QueryCellInfoEventArgs) {
 var args = QueryCellInfoEventArgs;
 if (args.column.field === 'Progress' && +args.cell.innerHTML > 90 &&
+args.cell.innerHTML <= 100) {
 args.cell.setAttribute('style', 'background-
color:#336c12;color:white;');
 } else if (+args.cell.innerHTML > 20 && args.column.field ===
'Progress') {
 args.cell.setAttribute('style', 'background-
color:#7b2b1d;color:white;');
 }
 }
</script>
```

### QUERYCELL.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

<!-- Auto wrap

The auto wrap allows the cell content of the treegrid to wrap to the next line when it exceeds the boundary of the cell width. The Cell Content wrapping works based on the position of white space between words.

To enable auto wrap, set the [allowTextWrap](#) property to **true**.

You can configure the auto wrap mode by setting the [textWrapSettings.wrapMode](#) property.



There are three types of `wrapMode`. They are:

- **Both:** `Both` value is set by default. Auto wrap will be enabled for both the content and the header.
- **Header:** Auto wrap will be enabled only for the header.
- **Content:** Auto wrap will be enabled only for the content.

When a column width is not specified, then auto wrap of columns will be adjusted with respect to the treegrid's width.

In the following example, the `textWrapSettings.wrapMode` is set to `Content`.

#### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(300)
 .AllowTextWrap(true)
 .TextWrapSettings(text => {
text.WrapMode(Syncfusion.EJ2.TreeGrid.WrapMode.Content); })
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render())
```

#### AUTOWRAP.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

#### Custom Attributes

You can customize the treegrid cells by adding a CSS class to the `customAttribute` property of the column.

```
`CSS
.e-attr {
background: '#d7f0f4';
}
```

In the below example, we have customized the cells of **TaskID** and **StartDate** columns.

#### CSHTML

```
<style>
 .e-attr {
 background: '#d7f0f4';
 }
</style>
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(300)
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).CustomAttributes(new { @class = "e-attr"
}).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Cust
omAttributes(new { @class = "e-attr" }).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Widt
h(90).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render())
```

#### CUSTOMATTR.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

#### Grid Lines

The [gridLines](#) have option to display cell border and it can be defined by the [gridLines](#) property.

The available modes of grid lines are:

- | Modes | Actions |
- |-----|-----|
- | Both | Displays both the horizontal and vertical grid lines. |
- | None | No grid lines are displayed. |
- | Horizontal | Displays the horizontal grid lines only. |
- | Vertical | Displays the vertical grid lines only. |
- | Default | Displays grid lines based on the theme. |

#### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(300)
 .GridLines(GridLine.Both)
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render())
```

### GRIDLINES.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

By default, the treegrid renders with **Default** mode.

#### Clip Mode

The clip mode provides options to display its overflow cell content and it can be defined by the [columns.clipMode](#) property.

There are three types of [clipMode](#). They are:

- **Clip**: Truncates the cell content when it overflows its area.
- **Ellipsis**: Displays ellipsis when the cell content overflows its area.
- **EllipsisWithTooltip**: Displays ellipsis when the cell content overflows its area, also it will display the tooltip while hover on ellipsis is applied.

### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(300)
 .GridLines(GridLine.Both)
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").Width(160).ClipMode(ClipMode.EllipsisWithTooltip).Add();
```

```
col.Field("Assignee.FirstName").HeaderText("Assignee").ClipMode(ClipMode.Ellipsis).Width(110).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).ClipMode(ClipMode.Clip).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render();
```

### CLIPMODE.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

By default, [columns.clipMode](#) value is **Ellipsis**.

-->

### Editing in ASP.NET MVC Tree Grid Component

The TreeGrid component has options to dynamically insert, delete and update records.

Editing feature is enabled by using [EditSettings](#) property and it requires a primary key column for CRUD operations.

To define the primary key, set [IsPrimaryKey](#) to **true** using [Column](#) API of that particular column.

### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(270)
 .EditSettings(edit =>
 edit.AllowAdding(true).AllowDeleting(true).AllowEditing(true))
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").IsPrimaryKey(true).Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("Priority").HeaderText("Priority").Width(90).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(110).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render())
```

### EDIT.CS

```
public IActionResult Index()
```

```
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can disable editing for a particular column, by specifying [AllowEditing](#) of [Column](#) API to **false**.

#### Toolbar with edit option

The treegrid toolbar has the built-in items to execute Editing actions.

You can define this by using the [Toolbar](#) property.

#### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(270)
 .EditSettings(edit =>
edit.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusio
n.EJ2.TreeGrid.EditMode.Row)
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update",
"Cancel" })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").IsPrimaryKey(true).Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("Priority").HeaderText("Priority").Width(90).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Widt
h(110).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render())
```

#### EDITTOOLS.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

#### Adding row position

The TreeGrid control provides the support to add the new row in the top, bottom, above selected row, below selected row and child position of tree grid content using [NewRowPosition](#) property of [EditSettings](#) API. By default, a new row will be added at the top of the treegrid.

The following examples shows how to set new row position as **Child** in tree grid.

#### CSHTML

```
@using Syncfusion.EJ2.Grids
```

```
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .EditSettings(edit =>
 {
 edit.AllowAdding(true);
 edit.AllowDeleting(true);
 edit.AllowEditing(true);
 edit.Mode(Syncfusion.EJ2.TreeGrid.EditMode.Cell);
 edit.NewRowPosition(Syncfusion.EJ2.TreeGrid.RowPosition.Child);
 })
 .Toolbar(new List<string>() { "Add", "Delete", "Update", "Cancel" })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").IsPrimaryKey(true).Width(120)
 .TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Add();
 col.Field("StartDate").HeaderText("Start Date").Width(150).Format("yMd")
 .EditType("datepickeredit").TextAlign(TextAlign.Right).Add();

 col.Field("Duration").HeaderText("Duration").Width("110").EditType("numericedit")
 .Edit(new { @params = new { format = "n" } })
 .TextAlign(TextAlign.Right).Add();
 })
 .Height(400).ChildMapping("Children").TreeColumnIndex(1).Render())
```

### NEWROW.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.dataSource = tree;
 return View();
}
```

### Confirmation messages

#### Delete confirmation

The delete confirm dialog can be shown when deleting a record by defining the [ShowDeleteConfirmDialog](#) as **true**

### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .EditSettings(edit =>
 {
 edit.AllowAdding(true);
 edit.AllowDeleting(true);
 edit.AllowEditing(true);
 edit.ShowDeleteConfirmDialog(true);
 edit.Mode(Syncfusion.EJ2.TreeGrid.EditMode.Cell);
 })
 .Toolbar(new List<string>() { "Add", "Update", "Delete", "Cancel" })
```

```

 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").IsPrimaryKey(true).Width(120)
 .TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Add();
 col.Field("StartDate").HeaderText("Start
Date").Width(150).Format("yMd")
 .EditType("datepickeredit").TextAlign(TextAlign.Right).Add();

 col.Field("Duration").HeaderText("Duration").Width("110").EditType("numeric
edit")
 .Edit(new { @params = new { format = "n" }
 }).TextAlign(TextAlign.Right).Add();
 }).Height(400).ChildMapping("Children").TreeColumnIndex(1).Render()
)

```

### **DELETECONFIRM.CS**

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.dataSource = tree;
 return View();
}

```

**Note:** The [ShowDeleteConfirmDialog](#) supports all type of edit modes.

Default column values on add new

The treegrid provides an option to set the default value for the columns when adding a new record in it.

To set a default value for the particular column by defining the [DefaultValue](#) in [Column](#) API.

### **CSHTML**

```

@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.dataSource)
 .EditSettings(edit =>
 {
 edit.AllowAdding(true);
 edit.AllowDeleting(true);
 edit.AllowEditing(true);
 edit.Mode(Syncfusion.EJ2.TreeGrid.EditMode.Cell);
 })
 .Toolbar(new List<string>() { "Add", "Delete", "Update", "Cancel" })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").IsPrimaryKey(true).Width(120)
 .TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Add();
 col.Field("StartDate").HeaderText("Start
Date").Width(150).Format("yMd")
 .EditType("datepickeredit").TextAlign(TextAlign.Right).Add();
 })

```

```
col.Field("Priority").HeaderText("Priority").Width("110").DefaultValue("Normal").Add();
 }).Height(400).ChildMapping("Children").TreeColumnIndex(1).Render()
)
```

### DEFAULTVALUE.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

### Disable editing for particular column

You can disable editing for particular columns by using the [AllowEditing](#) property of [Column](#) API.

In the following demo, editing is disabled for the **Start Date** column.

### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .EditSettings(edit =>
 {
 edit.AllowAdding(true);
 edit.AllowDeleting(true);
 edit.AllowEditing(true);
 edit.Mode(Syncfusion.EJ2.TreeGrid.EditMode.Cell);
 })
 .Toolbar(new List<string>() { "Add", "Delete", "Update", "Cancel" })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").IsPrimaryKey(true).Width(120)
 .TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Add();
 col.Field("StartDate").HeaderText("Start Date").Width(150).Format("yMd")
 .EditType("datepickeredit").TextAlign(TextAlign.Right).AllowEditing(false).Add();
 col.Field("Priority").HeaderText("Priority").Width("110").Add();
 })
).Height(400).ChildMapping("Children").TreeColumnIndex(1).Render()
)
```

### DISABLE.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.dataSource = tree;
 return View();
}
```



```
}

```

### Troubleshoot: Editing works only for first row

The Editing functionalities can be performed based upon the primary key value of the selected row.

If [IsPrimaryKey](#) is not defined in the treegrid, then edit or delete action take places the first row.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to knows how to present and manipulate data.

### Sorting

Sorting enables you to sort data in the **Ascending** or **Descending** order.

To sort a column, click the column header.

To sort multiple columns, press and hold the CTRL key and click the column header. You can clear sorting of any one of the multi-sorted columns by pressing and holding the SHIFT key and clicking the specific column header.

To enable sorting in the TreeGrid, set the [AllowSorting](#) to true. Sorting options can be configured through the [SortSettings](#).

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(315).AllowSorting().DataSource((IEnumerable<object>) ViewBag.datasource).Columns(col =>
{
 col.Field("category").HeaderText("Category").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(210).Add();
 col.Field("orderDate").HeaderText("Order Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("units").HeaderText("Units").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("subTasks").AllowPaging(true).TreeColumnIndex(1).Render()
```

### DEFAULTSORTING.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** TreeGrid columns are sorted in the **Ascending** order. If you click the already sorted column, the sort direction toggles.

<br/> You can apply and clear sorting by invoking [sortByColumn](#) and [clearSorting](#) methods.

<br/> To disable sorting for a particular column, set the [AllowSorting](#) property of [Column](#) to **false**.

### Initial sort

To sort at initial rendering, set the **Field** and **Direction** in the [Columns](#) property of [SortSettings](#).

#### CSHTML

```
@{
 List<object> cols = new List<object>();
 cols.Add(new { field = "category", direction = "Ascending" });
 cols.Add(new { field = "orderName", direction = "Descending" });
}
@Html.EJS().TreeGrid("TreeGrid").Height(315).AllowSorting().DataSource((IEnumerable<object>)ViewBag.datasource).SortSettings(sort =>
sort.Columns(cols)).Columns(col =>
{
 col.Field("category").HeaderText("Category").Width(110).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(210).Add();
 col.Field("orderDate").HeaderText("Order
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("units").HeaderText("Units").TextAlign(Syncfusion.EJ2.Grids.TextAl
ign.Right).Width(190).Add();
}).ChildMapping("subTasks").AllowPaging().TreeColumnIndex(1).Render()
```

#### INITIALSORT.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

### Sorting events

During the sort action, the treegrid component triggers two events. The [ActionBegin](#) event triggers before the sort action starts, and the [ActionComplete](#) event triggers after the sort action is completed. Using these events you can perform the needed actions.

#### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(315).AllowSorting().DataSource((IEnumerable<object>)ViewBag.datasource).ActionBegin("actionHandler").Columns(col
=>
{
 col.Field("category").HeaderText("Category").Width(110).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(210).Add();
 col.Field("orderDate").HeaderText("Order
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();
```

```
col.Field("units").HeaderText("Units").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("subTasks").AllowPaging(true).ActionComplete("actionHandler").TreeColumnIndex(1).Render()
<script>
 function actionHandler(args) {
 alert(args.requestType + ' ' + args.type); //custom Action
 }
</script>
```

### EVENTHANDLERS.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** The **args.requestType** is the current action name. For example, in sorting the **args.requestType** value is 'sorting'.

<!-- Custom sort comparer

You can customize the default sort action for a column by defining the [column.sortComparer](#) property. The sort comparer function has the same functionality like [Array.sort](#) sort comparer.

In the following example, custom sort comparer function was defined in the **Category** column.



```
`typescript
import { TreeGrid, Sort } from '@syncfusion/ej2-treegrid';
import { sortData } from './datasource.ts';
TreeGrid.Inject(Sort);
// The custom function
let sortComparer: (reference: string, comparer: string) => number = (reference: string,
comparer: string) => {
 if (reference < comparer) {
 return -1;
 }
 if (reference > comparer) {
 return 1;
 }
 return 0;
};
```

```
let treeGridObj: TreeGrid = new TreeGrid({
 dataSource: sortData,
 childMapping: 'subtasks',
 allowSorting: true,
 height: 315,
 treeColumnIndex: 1,
 columns: [
 { field: 'Category', headerText: 'Category', width: 140 },
 { field: 'orderName', headerText: 'Order Name', width: 200 },
 { field: 'orderDate', headerText: 'Order Date', width: 120, textAlign: 'Right', format: 'yMd', type: 'date' },
 { field: 'units', headerText: 'Units', width: 90, textAlign: 'Right' }
]
});
treeGridObj.appendTo('#TreeGrid');
```

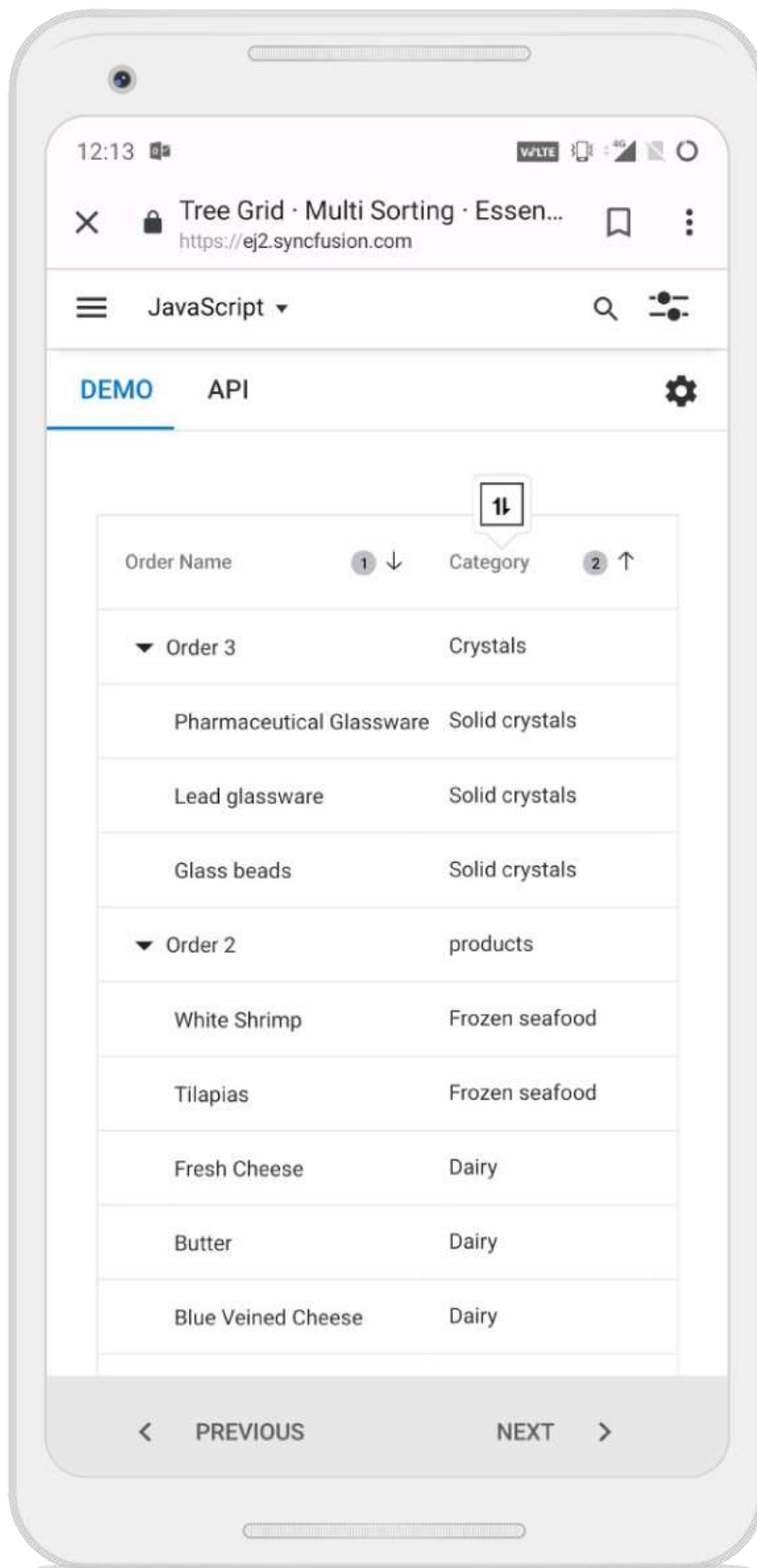
**Note:** The sort comparer function will work only for the local data. -->

#### Touch interaction

When you tap the treegrid header on touchscreen devices, the selected column header is sorted. A

popup  is displayed for multi-column sorting. To sort multiple columns, tap the popup , and then tap the desired treegrid headers.

The following screenshot shows treegrid touch sorting.



**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Filtering in ASP.NET MVC Tree Grid Component

Filtering allows you to view specific or related records based on filter criteria. To enable filtering in the TreeGrid, set the [AllowFiltering](#) to true. Filtering options can be configured through [FilterSettings](#).

#### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(275).AllowFiltering().DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").TreeColumnIndex(1).AllowPaging().Render()
```

#### DEFAULTFILTERING.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can apply and clear filtering by using [filterByColumn](#) and [clearFiltering](#) methods.

To disable filtering for a particular column, set [AllowFiltering](#) property of [Column](#) to false.

#### Filter hierarchy modes

TreeGrid provides support for a set of filtering modes with [HierarchyMode](#) of [FilterSettings](#) property.

The below are the type of filter mode available in TreeGrid.

- **Parent** : This is the default filter hierarchy mode in TreeGrid. The filtered records are displayed with its parent records, if the filtered records not have any parent record then the filtered records only displayed.
- **Child** : The filtered records are displayed with its child record, if the filtered records not have any child record then the filtered records only displayed.
- **Both** : The filtered records are displayed with its both parent and child record, if the filtered records not have any parent and child record then the filtered records only displayed.
- **None** : The filtered records are only displayed.

#### CSHTML

```

@Html.EJS().DropDownList("FilterMode").DataSource((IEnumerable<object>)ViewBag.dropdata).Value("Parent").Width("100").Fields(new
Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "id", Value =
"mode" }).Change("onChange").Render()
@Html.EJS().TreeGrid("TreeGrid").Height(275).AllowFiltering().DataSource((IE
numerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
<script>
 function onChange(e) {
 var mode = e.value;
 var treegrid = document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.filterSettings.hierarchyMode = mode;
 }
</script>

```

#### FILTERMODES.CS

```

public ActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 ViewBag.dropdata = new List<object>() {
 new { id= "Parent", mode= "Parent" },
 new { id= "Child", mode= "Child" },
 new { id= "Both", mode= "Both" },
 new { id= "None", mode= "None" },
 };
 return View();
}

```

#### Initial filter

To apply the filter at initial rendering, set the filter **Predicate** object in [Columns](#) property of [FilterSettings](#).

#### CSHTML

```

@{
 List<object> filterColumns = new List<object>();
 filterColumns.Add(new { field = "TaskName", matchCase = false, @operator
= "startswith", predicate = "and", value = "plan" });
 filterColumns.Add(new { field = "Duration", matchCase = false, @operator
= "equal", predicate = "and", value = 5 });
}
@Html.EJS().TreeGrid("TreeGrid").Height(275).AllowFiltering().FilterSettings
(filter =>

```

```
filter.Columns(filterColumns)).DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
```

### INITIALFILTER.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

### Filter operators

The filter operator for a column can be defined in the **Operator** property of [Columns](#) in [FilterSettings](#).

The available operators and its supported data types are:

Operator | Description | Supported Types

startswith | Checks whether the value begins with the specified value. | String

endswith | Checks whether the value ends with the specified value. | String

contains | Checks whether the value contains the specified value. | String

equal | Checks whether the value is equal to the specified value. | String &#124; Number &#124; Boolean &#124; Date

notequal | Checks for values not equal to the specified value. | String &#124; Number &#124; Boolean &#124; Date

greaterthan | Checks whether the value is greater than the specified value. | Number &#124; Date

greaterthanorequal | Checks whether a value is greater than or equal to the specified value. | Number &#124; Date

lessthan | Checks whether the value is less than the specified value. | Number &#124; Date

lessthanorequal | Checks whether the value is less than or equal to the specified value. | Number &#124; Date

**Note:** By default, the **Operator** [Columns](#) value is **equal**.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.



## Search

You can search records in a TreeGrid, by using the [search](#) method with search key as a parameter. This also provides an option to integrate search text box in treegrid's toolbar by adding [search](#) item to the [Toolbar](#).

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(270).Toolbar(new List<string>() {
 "Search" }).DataSource((IEnumerable<object>)ViewBag.datasources).Columns(col
=>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
```

### DEFAULTSEARCHING.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasources = tree;
 return View();
}
```

## Initial search

To apply search at initial rendering, set the fields, operator, key, and ignoreCase in the [SearchSettings](#).

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(270).Toolbar(new List<string>() {
 "Search"
}).DataSource((IEnumerable<object>)ViewBag.datasources).SearchSettings(search
=>
{
 search.Fields(new string[] { "TaskName" })
 .Operators("contains")
 .Key("plan")
 .IgnoreCase(true);
}).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();
}
```

```
col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").TreeColumnIndex(1).AllowPaging().Render()
```

### INITIALSEARCH.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** By default, treegrid searches all the bound column values. To customize this behavior define the [SearchSettings.Fields](#) property.

### Search operators

The search operator can be defined in the [Operators](#) property of [SearchSettings](#) to configure specific searching.

The following operators are supported in searching:

#### Operator | Description

startsWith | Checks whether a value begins with the specified value.

endsWith | Checks whether a value ends with the specified value.

contains | Checks whether a value contains the specified value.

equal | Checks whether a value is equal to the specified value.

notEqual | Checks for values not equal to the specified value.

**Note:** By default, the [Operators](#) value is **contains**.

### Search by external button

To search treegrid records from an external button, invoke the [search](#) method.

### CSHTML

```
@Html.TextBox("searchText")
@Html.EJS().Button("Search").Content("Search").CssClass("e-
primary").Render()
@Html.EJS().TreeGrid("TreeGrid").Height(270).DataSource((IEnumerable<object>
)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
```

```
<script>
 document.getElementById('Search').addEventListener('click', function
 (args) {
 var val = document.getElementById('searchText').value;
 var treegrid = document.getElementById('TreeGrid').ej2_instances[0];
 treegrid.search(val);
 });
</script>
```

### SEARCHEXTERNAL.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

### Search specific columns

By default, treegrid searches all visible columns. You can search specific columns by defining the specific column's field names in the [Fields](#) property of [SearchSettings](#).

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").Height(270).Toolbar(new List<string>() {
 "Search"
}).DataSource((IEnumerable<object>)ViewBag.datasource).SearchSettings(search
=>
{
 search.Fields(new string[] { "TaskName", "Duration" });
}).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
```

### SEARCHCOLUMNS.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Paging

Paging provides an option to display TreeGrid data in page segments. To enable paging, set the [AllowPaging](#) to true. When paging is enabled, pager component renders at the bottom of the treegrid.

Paging options can be configured through the [PageSettings](#).

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").AllowPaging().DataSource((IEnumerable<object>)ViewBag.datasource).PageSettings(page => page.PageSize(7)).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
```

### DEFAULTPAGING.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can achieve better performance by using treegrid paging to fetch only a pre-defined number of records from the data source.

### Page Size Mode

Two behaviours are available in TreeGrid paging to display certain number of records in a current page. Following are the two types of [PageSizeMode](#) property of [PageSettings](#).

- **All** : This is the default mode. The number of records in a page is based on [PageSize](#) property.
- **Root** : The number of root nodes or the 0th level records to be displayed per page is based on [PageSize](#) property.

With [PageSizeMode](#) property as **Root**, only the root level or the 0th level records are considered in records count.

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").AllowPaging().DataSource((IEnumerable<object>)ViewBag.datasource).PageSettings(page =>
```

```

page.PageSize(2).PageSizeMode(Syncfusion.EJ2.TreeGrid.PageSizeMode.Root)).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()

```

### PAGEMODE.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Template

You can use custom elements inside the pager instead of default elements.

The custom elements can be defined by using the [Template](#) property.

Inside this template, you can access the [CurrentPage](#), [PageSize](#), [PageCount](#), [TotalPage](#) and [TotalRecordCount](#) values.

### CSHTML

```

<script id="template" type="text/x-template">
 <div class="e-pagertemplate">
 <div class="col-lg-12 control-section">
 <div class="content-wrapper">
 <input id="currentPage" type="text" value=${currentPage}
style="padding-left: 10px; text-align: left">
 </div>
 <div id="totalPages" class="e-pagertemplatemessage" style="margin-top: 5px; margin-left: 30px; border: none; display: inline-block">
 ${currentPage} of ${totalPages} pages
 (${totalRecordsCount} items)
 </div>
 </div>
 </script>
@Html.EJS().TreeGrid("TreeGrid").AllowPaging().DataSource((IEnumerable<object>)ViewBag.datasource).PageSettings(page =>
page.PageSize(6).Template("#template")).ActionComplete("actionComplete").DataBound("dataBound").Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
}

```

```

 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
}).ChildMapping("Children").TreeColumnIndex(1).AllowPaging().Render()
<script>
 var flag = true;
 function dataBound(args) {
 if (flag) {
 flag = false;
 updateTemplate();
 }
 }
 function actionComplete(args) {
 if (args.requestType === 'paging') {
 updateTemplate();
 }
 }
 function updateTemplate() {
 var numeric = new ej.inputs.NumericTextBox({
 min: 1,
 max: 6,
 step: 1,
 width: 75,
 format: '###.##',
 change: function (args) {
 var treegridObj =
document.getElementById('TreeGrid').ej2_instances[0];
 var value = args.value;
 treegridObj.goToPage(value);
 }
 });
 numeric.appendTo('#currentPage');
 };
</script>

```

### PAGERTEMPLATE.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Pager with Page Size Dropdown

The pager Dropdown allows you to change the number of records in the TreeGrid dynamically. It can be enabled by defining the [PageSizes](#) property of [PageSettings](#) as **true**.

### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").AllowPaging().DataSource((IEnumerable<object>)ViewBag.datasource).PageSettings(page =>
page.PageSize(10).PageSizes(true)).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();
 col.Field("EndDate").HeaderText("End Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(110).Add();

 col.Field("Progress").HeaderText("Progress").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(90).Add();
 col.Field("Priority").HeaderText("Priority").Width(90).Add();
}).ChildMapping("Children").TreeColumnIndex(1).Render()
```

### PAGESIZES.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

| Task ID                                                          | Task Name                  | Start Date | End Date  | Duration | Progress | Priority                |
|------------------------------------------------------------------|----------------------------|------------|-----------|----------|----------|-------------------------|
| 1                                                                | ▼ Planning                 | 2/3/2017   | 2/7/2017  | 5        | 100      | Normal                  |
| 2                                                                | Plan timeline              | 2/3/2017   | 2/7/2017  | 5        | 100      | Normal                  |
| 3                                                                | Plan budget                | 2/3/2017   | 2/7/2017  | 5        | 100      | Low                     |
| 4                                                                | Allocate resources         | 2/3/2017   | 2/7/2017  | 5        | 100      | Critical                |
| 5                                                                | Planning complete          | 2/7/2017   | 2/7/2017  | 0        | 0        | Low                     |
| 6                                                                | ▼ Design                   | 2/10/2017  | 2/14/2017 | 3        | 86       | High                    |
| 7                                                                | Software Specification     | 2/10/2017  | 2/12/2017 | 3        | 60       | Normal                  |
| 8                                                                | Develop prototype          | 2/10/2017  | 2/12/2017 | 3        | 100      | Critical                |
| 9                                                                | Get approval from customer | 2/13/2017  | 2/14/2017 | 2        | 100      | Low                     |
| 10                                                               | Design Documentation       | 2/13/2017  | 2/14/2017 | 2        | 100      | High                    |
| <div> <div> 1 2 ... 10 </div> <div> Items per page </div> </div> |                            |            |           |          |          | 1 of 4 pages (36 items) |

### How to render Pager at the Top of the TreeGrid

By default, Pager will be rendered at the bottom of the TreeGrid. You can also render the Pager at the top of the TreeGrid by using the [DataBound](#) event.

#### CSHTML

```
@Html.EJS().TreeGrid("TreeGrid").AllowPaging().DataSource((IEnumerable<object>)ViewBag.datasource).PageSettings(page =>
page.PageSize(10).PageSizes(true)).DataBound("dataBound").Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();
 col.Field("EndDate").HeaderText("End
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(110).Add();

 col.Field("Progress").HeaderText("Progress").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(90).Add();
 col.Field("Priority").HeaderText("Priority").Width(90).Add();
}).ChildMapping("Children").AllowPaging().TreeColumnIndex(1).Render()
<script>
 var initialLoad = true;
 function dataBound(args) {
 if (initialLoad) {
 initialLoad = false;
 var pager = document.getElementsByClassName('e-gridpager');
 var topElement;
 if (this.toolbar) {
 topElement = document.getElementsByClassName('e-toolbar');
 } else {
 topElement = document.getElementsByClassName('e-
gridheader');
 }
 topElement[0].before(pager[0]);
 }
 };
</script>
```

#### CUSTOMIZE.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** During the paging action, the pager component triggers the below three events.



<br/> The **created** event triggers when Pager is created.

<br/> The **click** event triggers when the numeric items in the pager is clicked.

<br/> The **dropDownChanged** event triggers when pageSize DropDownList value is selected.

<br/> You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Selection in ASP.NET MVC Tree Grid Component

Selection provides an option to highlight a row or a cell. It can be done through simple mouse down or arrow keys. To disable selection in the TreeGrid, set the [AllowSelection](#) to false.

The treegrid supports two types of selection that can be set by using the [Type](#) property in [SelectionSettings](#). They are:

- **Single:** The Single value is set by default, and it only allows selection of a single row or a cell.
- **Multiple:** Allows you to select multiple rows or cells.

To perform the multi-selection, press and hold CTRL key and click the desired rows or cells. To select range of rows or cells, press and hold the SHIFT key and click the rows or cells.

#### CSHTML

```
@(Html.EJS().TreeGrid("SelectionAPI").AllowPaging().DataSource((IEnumerable<
object>) ViewBag.datasource)
 .AllowSelection().Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
 })
 .ChildMapping("Children").SelectionSettings(select =>
select.Type(Syncfusion.EJ2.Grids.SelectionType.Multiple)).TreeColumnIndex(1)
 .Render())
```

#### SELECTION.CS

```
public ActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

### Selection mode

The treegrid supports three types of selection mode that can be set by using the [Mode](#) property of [SelectionSettings](#). They are:

- **Row:** The Row value is set by default, and allows you to select only rows.
- **Cell:** Allows you to select only cells.
- **Both:** Allows you to select rows and cells at the same time.

### CSHTML

```
@(Html.EJS().TreeGrid("SelectionAPI").AllowPaging().DataSource((IEnumerable<
object>)ViewBag.datasource)
 .AllowSelection()
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();
 col.Field("StartDate").HeaderText("Start
Date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Format("yMd").Width(2
10).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(Syncfusion.EJ2.Grids.
TextAlign.Right).Width(190).Add();
 })
 .ChildMapping("Children").SelectionSettings(select =>
select.Mode(Syncfusion.EJ2.Grids.SelectionMode.Both)).TreeColumnIndex(1).Ren
der())
```

### SELECTION-MODE.CS

```
public ActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Aggregates in ASP.NET MVC Tree Grid Component

Aggregate values are displayed in the TreeGrid footer and in parent row footer for child row aggregate values. It can be configured through [Aggregates](#) property.

[Field](#) and [Type](#) are the minimum properties required to represent an aggregate column.

By default, the aggregate value can be displayed in the treegrid footer, and footer of child rows. To show the aggregate value in one of the cells, use the [FooterTemplate](#).

### Built-in aggregate types

The aggregate type should be specified in the [Type](#) property to configure an aggregate column.

The built-in aggregates are,

- Sum
- Average
- Min
- Max
- Count
- Truecount
- Falsecount

**Note:** Multiple aggregates can be used for an aggregate column by setting the [Type](#) property with an array of aggregate types.

<br/> Multiple types for a column is supported only when one of the aggregate templates is used.

#### Child aggregate

Aggregate value is calculated for child rows, and it is displayed in the parent row footer. Use the [ShowChildSummary](#) property to render the child rows aggregate value.

#### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid")
 .Height(260)
 .DataSource((IEnumerable<object>)ViewBag.datasources)
 .Aggregates(agg =>
 {
 agg.Columns(new
List<Syncfusion.EJ2.TreeGrid.TreeGridAggregateColumn>() {
 new Syncfusion.EJ2.TreeGrid.TreeGridAggregateColumn() {
 Field = "units", Type = "Max",
 FooterTemplate = "Maximum: ${Max}" , ColumnName = "units"
 },
 new Syncfusion.EJ2.TreeGrid.TreeGridAggregateColumn() {
 Field = "price", Type = "Min",
 FooterTemplate = "Minimum: ${Min}"
 }
 })
 .ShowChildSummary(true).Add();
 })
 .Columns(col =>
 {
 col.Field("orderID").HeaderText("Order
ID").Width(130).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(195).Add();

 col.Field("units").HeaderText("Units").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(130).Add();

 col.Field("price").HeaderText("Price").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(125).Add();
 })
 .ChildMapping("subTasks").TreeColumnIndex(1).Render())
```

#### CHILDAGGREGATE.CS

```
public IActionResult Index()
{
```

```

ViewBag.datasource = TreeSummaryData.GetData();
return View();
}

```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Context menu in ASP.NET MVC Tree Grid Component

The TreeGrid has options to show the context menu when right clicked on it. To enable this feature, you need to define either default or custom item in the [ContextMenuItems](#).

The default items are in the following table.

Items | Description

**AutoFit** | Auto fit the current column.

**AutoFitAll** | Auto fit all columns.

**Edit** | Edit the current record.

**Delete** | Delete the current record.

**Save** | Save the edited record.

**Cancel** | Cancel the edited state.

**PdfExport** | Export the treegrid data as Pdf document.

**ExcelExport** | Export the treegrid data as Excel document.

**CsvExport** | Export the treegrid data as CSV document.

**SortAscending** | Sort the current column in ascending order.

**SortDescending** | Sort the current column in descending order.

**FirstPage** | Go to the first page.

**PrevPage** | Go to the previous page.

**LastPage** | Go to the last page.

**NextPage** | Go to the next page.

**AddRow** | Add new row to the treegrid.

### CSHTML

```

@(Html.EJS().TreeGrid("TreeGrid").AllowPaging().AllowSorting().AllowExcelExport()
 .AllowPdfExport()
 .AllowResizing()
 .EditSettings(edit =>
edit.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusion.EJ2.TreeGrid.EditMode.Row))
 .PageSettings(page => page.PageSize(7))
 .DataSource((IEnumerable<object>) ViewBag.datasource)

```

```

 .ContextMenuItems (
 new List<object>() { "SortAscending",
 "SortDescending", "Edit", "Delete",
 "Save", "Cancel", "PdfExport", "ExcelExport", "CsvExport",
 "FirstPage", "PrevPage", "LastPage", "NextPage", "Indent",
 "Outdent"
 })
 .Columns (col =>
 {
 col.Field("TaskId").HeaderText("Task ID").Width(90).IsPrimaryKey(true).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start Date").Format("yMd").Type("date").EditType("datepickeredit").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(90).Add();
 col.Field("Duration").HeaderText("Duration").Width(80).EditType("numericedit").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 })
 .ChildMapping("Children").SelectedRowIndex(2).TreeColumnIndex(1).Render()

```

### DEFAULT-CONTEXTMENU.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Custom context menu items

The custom context menu items can be added by defining the [ContextMenuItems](#) as a collection of **ContextMenuitemModel**.

Actions for this customized items can be defined in the [ContextMenuClick](#) event.

In the below sample, we have shown context menu item for parent rows to expand or collapse child rows.

### CSHTML

```

@{
 List<object> ContextMenuItems = new List<object>();
 ContextMenuItems.Add(new { text = "Collapse the Row", target = ".e-content", id = "collapserow" });
 ContextMenuItems.Add(new { text = "Expand the Row", target = ".e-content", id = "expandrow" });
}
@ (Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {

```

```

 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Width(90).Add();

col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();

}).ChildMapping("Children").ContextMenuItems(ContextMenuItems).ContextMenuOp
en("ContextMenuOpen").ContextMenuClick("ContextMenuClick")
 .TreeColumnIndex(1).Render();
<script>
 function ContextMenuOpen(args) {
 var treegrid = document.getElementById("TreeGrid").ej2_instances[0];
 var elem = args.event.target;
 var uid = elem.closest('.e-row').getAttribute('data-uid');
 if (ej.base.isNullOrUndefined(ej.base.getValue('hasChildRecords',
treegrid.grid.getRowObjectFromUID(uid).data))) {
 args.cancel = true;
 } else {
 var flag = ej.base.getValue('expanded',
treegrid.grid.getRowObjectFromUID(uid).data);
 var val = flag ? 'none' : 'block';

document.querySelector('li#expandrow')[0].setAttribute('style', 'display:
' + val + ';');
 val = !flag ? 'none' : 'block';

document.querySelector('li#collapserow')[0].setAttribute('style',
'display: ' + val + ';');
 }
 }
 function ContextMenuClick(args) {
 var treegrid = document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.getColumnByField('TaskID');
 if (args.item.id === 'collapserow') {
 treegrid.collapseRow((treegrid.getSelectedRows()[0]),
treegrid.getSelectedRecords()[0]);
 } else {
 treegrid.expandRow((treegrid.getSelectedRows()[0]),
treegrid.getSelectedRecords()[0]);
 }
 }
}
</script>

```

### CUSTOM-CONTEXTMENU.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Enable and disable context menu items dynamically

You can enable and disable the context menu items using the [enableItems](#) method in [contextMenuOpen](#) event.

#### CSHTML

```
@model List<TreeGridSample.Controllers.TreeGridItems>
@{
 List<object> ContextMenuItems = new List<object>();
 ContextMenuItems.Add(new { text= "Edit Record", target= ".e-content",
id= "Edit_record" });
 ContextMenuItems.Add(new { text= "Delete Record", target= ".e-content",
id= "Delete_record" });
}
@(Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>)Model)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Width(90).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.
EJ2.Grids.TextAlign.Right).Add();
 })
 .ChildMapping("Children")
 .EditSettings(edit =>
 edit.AllowAdding(true).AllowDeleting(true).AllowEditing(true).Mode(Syncfusio
n.EJ2.TreeGrid.EditMode.Row)
 .ContextMenuItems(ContextMenuItems)
 .ContextMenuOpen("ContextMenuOpen")
 .ContextMenuClick("ContextMenuClick")
 .TreeColumnIndex(1).Render())
 <script>
 function ContextMenuOpen(args) {
 if (args.rowInfo.rowData.hasChildRecords == true) {
 this.grid.contextMenuModule.contextMenu.enableItems(['Edit
Record'], true); //Enable edit
 this.grid.contextMenuModule.contextMenu.enableItems(['Delete
Record'], false); //Disable delete
 } else {
 this.grid.contextMenuModule.contextMenu.enableItems(['Edit
Record'], false); //Disable edit
 this.grid.contextMenuModule.contextMenu.enableItems(['Delete
Record'], true); //Enable delete
 }
 }
 function ContextMenuClick(args) {
 if (args.element.innerHTML == "Edit Record") {
 this.startEdit(args.rowInfo.row);
 }
 else if (args.element.innerHTML == "Delete Record") {
 this.deleteRecord(args.rowInfo.row);
 }
 }
 }
}
```

```
</script>
```

### ENABLE-CONTEXTMENU.CS

```
public class HomeController : Controller
{
 public ActionResult Index()
 {
 return View(TreeGridItems.GetTreeData());
 }
}

public class TreeGridItems
{
 public TreeGridItems() { }
 public int TaskId { get; set; }
 public string TaskName { get; set; }
 public DateTime StartDate { get; set; }
 public int Duration { get; set; }
 public List<TreeGridItems> Children { get; set; }
 public static List<TreeGridItems> GetTreeData()
 {
 List<TreeGridItems> BusinessObjectCollection = new
List<TreeGridItems>();
 TreeGridItems Record1 = null;
 Record1 = new TreeGridItems()
 {
 TaskId = 1,
 TaskName = "Planning",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5,
 Children = new List<TreeGridItems>(),
 };
 TreeGridItems Child1 = new TreeGridItems()
 {
 TaskId = 2,
 TaskName = "Plan timeline",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child2 = new TreeGridItems()
 {
 TaskId = 3,
 TaskName = "Plan budget",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 TreeGridItems Child3 = new TreeGridItems()
 {
 TaskId = 4,
 TaskName = "Allocate resources",
 StartDate = new DateTime(2016, 06, 07),
 Duration = 5
 };
 Record1.Children.Add(Child1);
 Record1.Children.Add(Child2);
 Record1.Children.Add(Child3);
 }
}
```



```

TreeGridItems Record2 = new TreeGridItems()
{
 TaskId = 6,
 TaskName = "Design",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3,
 Children = new List<TreeGridItems>()
};
TreeGridItems Child5 = new TreeGridItems()
{
 TaskId = 7,
 TaskName = "Software Specification",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
};
TreeGridItems Child6 = new TreeGridItems()
{
 TaskId = 8,
 TaskName = "Develop prototype",
 StartDate = new DateTime(2021, 08, 25),
 Duration = 3
};
TreeGridItems Child7 = new TreeGridItems()
{
 TaskId = 9,
 TaskName = "Get approval from customer",
 StartDate = new DateTime(2024, 06, 27),
 Duration = 2
};
Record2.Children.Add(Child5);
Record2.Children.Add(Child6);
Record2.Children.Add(Child7);
BusinessObjectCollection.Add(Record1);
BusinessObjectCollection.Add(Record2);
return BusinessObjectCollection;
}
}

```

**Note:** You can hide or show an item in context menu for specific area inside of treegrid by defining the [Target](#) property.

You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### ToolBar in ASP.NET MVC Tree Grid Component

The TreeGrid provides ToolBar support to handle treegrid actions. The [ToolBar](#) property accepts either the collection of built-in toolbar items and [ItemModel](#) objects for custom toolbar items or HTML element ID for toolbar template.

#### Enable/disable toolbar items

You can enable/disable toolbar items by using the **enableItems** method.

#### CSHTML

```
@{
```

```

 List<object> toolbarItems = new List<object>();
 toolbarItems.Add("QuickFilter");
 toolbarItems.Add("ClearFilter");
 }
 @Html.EJS().Button("enable").Content("Enable").Render()
 @Html.EJS().Button("disable").Content("Disable").Render()
 @(Html.EJS().TreeGrid("ToolbarTemplate").DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Type("date").Width(90).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

 })
 .Height(200).ToolBar(toolbarItems).ToolBarClick("onToolBarClick").AllowFiltering()
 .ChildMapping("Children").TreeColumnIndex(1).Render())
<script>
 document.getElementById("enable").addEventListener('click', () => {
 var treegrid =
document.getElementById("ToolbarTemplate").ej2_instances[0];
 treegrid.toolbarModule.enableItems([treegrid.element.id +
'_gridcontrol_QuickFilter', treegrid.element.id +
'_gridcontrol_ClearFilter'], true); // enable toolbar items.
 });
 document.getElementById("disable").addEventListener('click', () => {
 var treegrid =
document.getElementById("ToolbarTemplate").ej2_instances[0];
 treegrid.toolbarModule.enableItems([treegrid.element.id +
'_gridcontrol_QuickFilter', treegrid.element.id +
'_gridcontrol_ClearFilter'], false); // disable toolbar items.
 });
 function onToolBarClick(args) {
 if (args.item.text === 'QuickFilter') {
 var treegrid =
document.getElementById("ToolbarTemplate").ej2_instances[0];
 treegrid.filterByColumn("TaskName", "startswith", "Testing");
 }
 if (args.item.text === 'ClearFilter') {
 var treegrid =
document.getElementById("ToolbarTemplate").ej2_instances[0];
 treegrid.clearFiltering();
 }
 }
}
</script>

```

### TOOLBAR-ENABLE.CS

```

public IActionResult Index()
{

```

```
var tree = TreeData.GetDefaultData();
ViewBag.datasource = tree;
return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

## Globalization in ASP.NET MVC Tree Grid control

### Localization

The [Localization](#) library allows you to localize default text content of the TreeGrid. The treegrid component has static text on some features (like toolbar area text, filter menu text, pager information text, etc.) that can be changed to other cultures (Arabic, Deutsch, French, etc.) by defining the [Locale](#) value and translation object.

The following list of properties and its values are used in the treegrid.

| Locale keywords | Text |

|-----|-----|

EmptyRecord | No records to display

True | true

False | false

ExpandAll | Expand All

CollapseAll | Collapse All

RowIndent | Indent

RowOutdent | Outdent

InvalidFilterMessage | Invalid Filter Data

FilterbarTitle | \s filter bar cell

Add | Add

Edit | Edit

Cancel | Cancel

Update | Update

Delete | Delete

Print | Print

Pdfexport | PDF Export

Excelexport | Excel Export

Wordexport | Word Export

Csvexport | CSV Export

Search | Search

Save | Save

EditOperationAlert | No records selected for edit operation

DeleteOperationAlert | No records selected for delete operation

SaveButton | Save

OKButton | OK

CancelButton | Cancel

EditFormTitle | Details of

AddFormTitle | Add New Record

ConfirmDelete | Are you sure you want to Delete Record?

SearchColumns | search columns

Matches | No Matches Found

FilterButton | Filter

ClearButton | Clear

StartsWith | Starts With

EndsWith | Ends With

Contains | Contains

Equal | Equal

NotEqual | Not Equal

LessThan | Less Than

LessThanOrEqual | Less Than Or Equal

GreaterThan | Greater Than

GreaterThanOrEqual | Greater Than Or Equal

ChooseDate | Choose a Date

EnterValue | Enter the value

autoFitAll | Auto Fit all columns

autoFit | Auto Fit this column

Export | Export

FirstPage | First Page

LastPage | Last Page

PreviousPage | Previous Page

NextPage | Next Page

SortAscending | Sort Ascending

SortDescending | Sort Descending

EditRecord | Edit Record

DeleteRecord | Delete Record

Above | Above

Below | Below

AddRow | Add Row

FilterMenu | Filter

SelectAll | Select All

Blanks | Blanks

FilterTrue | True

FilterFalse | False

NoResult | No Matches Found

ClearFilter | Clear Filter

NumberFilter | Number Filters

TextFilter | Text Filters

DateFilter | Date Filters

MatchCase | Match Case

Between | Between

CustomFilter | Custom Filter

CustomFilterPlaceholder | Enter the value

CustomFilterDatePlaceholder | Choose a date

AND | AND

OR | OR

ShowRowsWhere | Show rows where:

currentPageInfo | {0} of {1} pages

totalItemsInfo | ({0} items)

firstPageTooltip | Go to first page

lastPageTooltip | Go to last page

nextPageTooltip | Go to next page

previousPageTooltip | Go to previous page

nextPagerTooltip | Go to next pager items

previousPagerTooltip | Go to previous pager items

pagerDropDown | Items per page

pagerAllDropDown | Items

All | All

### Loading translations

To load translation object in an application, use **load** function of the **L10n** class.

The following example demonstrates the TreeGrid in **Deutsch** culture.

### CSHTML

```
@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.data
 source)
 .AllowFiltering()
 .AllowPaging()
 .PageSettings(page => page.PageSize(7))
 .FilterSettings(filter =>
 {
 filter.Type(Syncfusion.EJ2.TreeGrid.FilterType.Menu);
 })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(TextAlign.R
 ight).Add();
 })
 .Height(220).ChildMapping("Children").Toolbar(new List<string>() {
 "Print" })
 .Locale("de-DE").TreeColumnIndex(1).Render()
)
<script>
 ej.base.L10n.load({
 'de-DE': {
 'treegrid': {
 'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
 'Expand All': 'Alle erweitern',
 'Collapse All': 'Alles einklappen',
 'Print': "Drucken",
 'Pdfexport': "PDF-Export",
 'Excelexport': "Excel-Export",
 'Wordexport': "Word-Export",
 'FilterButton': "Filter",
 'ClearButton': "klar",
 'StartsWith': "Beginnt mit",
 'EndsWith': "Endet mit",
 'Contains': "Enthält",
 'Equal': "Gleich",
 'NotEqual': "Nicht gleich",
 'LessThan': "Weniger als",
 'LessThanOrEqual': "Weniger als oder gleich",
 'GreaterThan': "Größer als",
 'GreaterThanOrEqual': "Größer als oder gleich",
 'EnterValue': "Geben Sie den Wert ein",
 'FilterMenu': "Filter"
 },
 'pager': {
 'currentPageInfo': '{0} von {1} Seiten',
```

```

 'totalItemsInfo': '({0} Beiträge)',
 'firstPageTooltip': 'Zur ersten Seite',
 'lastPageTooltip': 'Zur letzten Seite',
 'nextPageTooltip': 'Zur nächsten Seite',
 'previousPageTooltip': 'Zurück zur letzten Seit',
 'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-
Elementen',
 'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
 },
 "dropdowns": {
 "noRecordsTemplate": "Keine Aufzeichnungen gefunden"
 },
 "datepicker": {
 "placeholder": "Wählen Sie ein Datum",
 "today": "heute"
 }
}
});
</script>

```

### GLOBAL.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Internationalization

The [Internationalization](#) library is used to globalize number, date, and time values in treegrid component using format strings in the [Format](#) property of [Column](#).

### CSHTML

```

@using Syncfusion.EJ2.Grids
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .AllowFiltering()
 .AllowPaging()
 .PageSettings(page => page.PageSize(7))
 .FilterSettings(filter =>
 {
 filter.Type(Syncfusion.EJ2.TreeGrid.FilterType.Menu);
 })
 .Columns(col =>
 {
 col.Field("orderId").HeaderText("Order ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("orderName").HeaderText("Order Name").Width(180).Add();

 col.Field("price").HeaderText("Price").Width(80).Format("C2").Type("number")
 .TextAlign(TextAlign.Right).Add();
 })
)

```

```

 }).Height(220).ChildMapping("subTasks").Toolbar(new List<string>() {
"Print" })
 .Locale("de-DE").TreeColumnIndex(1).Render()
)
<script>
 ej.base.setCurrencyCode('EUR');
 ej.base.L10n.load({
 'de-DE': {
 'treegrid': {
 'EmptyRecord': 'Keine Aufzeichnungen angezeigt',
 'Expand All': 'Alle erweitern',
 'Collapse All': 'Alles einklappen',
 'Print': "Drucken",
 'Pdfexport': "PDF-Export",
 'Excelexport': "Excel-Export",
 'Wordexport': "Word-Export",
 'FilterButton': "Filter",
 'ClearButton': "klar",
 'StartsWith': "Beginnt mit",
 'EndsWith': "Endet mit",
 'Contains': "Enthält",
 'Equal': "Gleich",
 'NotEqual': "Nicht gleich",
 'LessThan': "Weniger als",
 'LessThanOrEqual': "Weniger als oder gleich",
 'GreaterThan': "Größer als",
 'GreaterThanOrEqual': "Größer als oder gleich",
 'EnterValue': "Geben Sie den Wert ein",
 'FilterMenu': "Filter"
 },
 'pager': {
 'currentPageInfo': '{0} von {1} Seiten',
 'totalItemsInfo': '({0} Beiträge)',
 'firstPageTooltip': 'Zur ersten Seite',
 'lastPageTooltip': 'Zur letzten Seite',
 'nextPageTooltip': 'Zur nächsten Seite',
 'previousPageTooltip': 'Zurück zur letzten Seit',
 'nextPagerTooltip': 'Gehen Sie zu den nächsten Pager-
Elementen',
 'previousPagerTooltip': 'Gehen Sie zu vorherigen Pager-
Elementen'
 },
 'dropdowns': {
 'noRecordsTemplate': "Keine Aufzeichnungen gefunden"
 },
 'datepicker': {
 'placeholder': "Wählen Sie ein Datum",
 'today': "heute"
 }
 }
 });
</script>

```

### INTERNALIZATION.CS

```
public IActionResult Index()
```



```
{
 var tree = TreeDataFormat.GetDataFormat();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** In the above sample, **Price** column is formatted by **NumberFormatOptions**.

By default, [Locale](#) value is **en-US**. If you want to change the **en-US** culture to a different culture, you have to change the [Locale](#) accordingly.

### Right to left (RTL)

RTL provides an option to switch the text direction and layout of the TreeGrid component from right to left. It improves the user experiences and accessibility for users who use right-to-left languages (Arabic, Farsi, Urdu, etc.). To enable RTL Grid, set the [EnableRtl](#) to true.

### CSHTML

```
@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .AllowFiltering()
 .AllowPaging()
 .PageSettings(page => page.PageSize(7))
 .FilterSettings(filter =>
 {
 filter.Type(Syncfusion.EJ2.TreeGrid.FilterType.Menu);
 })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();

 col.Field("Duration").HeaderText("Duration").Width(80).TextAlign(TextAlign.Right).Add();
 }).Height(220).ChildMapping("Children").Toolbar(new List<string>() { "Print" })
 .Locale("ar-AE").EnableRtl(true).TreeColumnIndex(1).Render()
)
<script>
 ej.base.L10n.load({
 'ar-AE': {
 'treegrid': {
 "EmptyRecord": "لا سجلات لعرضها",
 "Print": "طباعة",
 "FilterButton": "منقي",
 "ClearButton": "واضح",
 "StartsWith": "ابدا ب",
 "EndsWith": "ينتهي مع",
 "Contains": "يحتوي على",
 "Equal": "مساو",
 "NotEqual": "غير متساوي",
 "LessThan": "أقل من",
 "LessThanOrEqual": "اصغر من او يساوي",
 "GreaterThan": "أكثر من",
 }
 }
 });
```

```

 "GreaterThanOrEqual": "أكبر من أو يساوي",
 "ChooseDate": "اختر تاريخا",
 "EnterValue": "أدخل القيمة",
 "FilterMenu": "منقي"
 },
 'pager': {
 'currentPageInfo': '{0} صفحة 1 {من}',
 'totalItemsInfo': '({0} العناصر)',
 'firstPageTooltip': 'انتقل إلى الصفحة الأولى',
 'lastPageTooltip': 'انتقل إلى الصفحة الأخيرة',
 'nextPageTooltip': 'انتقل إلى الصفحة التالية',
 'previousPageTooltip': 'انتقل إلى الصفحة السابقة',
 'nextPagerTooltip': 'انتقل إلى عناصر بيكر التالية',
 'previousPagerTooltip': 'للذهاب إلى عناصر بيكر السابقة'
 },
 "dropdowns": {
 "noRecordsTemplate": "لا توجد سجلات"
 },
 "datepicker": {
 "placeholder": "اختر تاريخا",
 "today": "اليوم"
 }
 }
 });
</script>

```

#### ENABLERTL.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

#### See Also

- [Internationalization](#)
- [Localization](#)

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

#### Scrolling

The scrollbar will be displayed in the treegrid when content exceeds the element [Width](#) or [Height](#). The vertical and horizontal scrollbars will be displayed based on the following criteria:

The vertical scrollbar appears when the total height of rows present in the treegrid exceeds its element height.

The horizontal scrollbar appears when the sum of columns width exceeds the treegrid element width.

The [Height](#) and [Width](#) are used to set the treegrid height and width, respectively.

**Note:** The default value for [Height](#) and [Width](#) is **auto**.

### Set width and height

To specify the [Width](#) and [Height](#) of the scroller in the pixel, set the pixel value to a number.

#### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
 })
 .Height(315).Width(400).ChildMapping("Children").TreeColumnIndex(1).Render
())
```

#### WIDTH-HEIGHT.CS

```
public IActionResult Index()
{
 var tree = TreeGridItems.GetTreeData();
 ViewBag.datasource = tree;
 return View();
}
```

### Responsive with parent container

Specify the [Width](#) and [Height](#) as **100%** to make the treegrid element fill its parent container.

Setting the [Height](#) to **100%** requires the treegrid parent element to have explicit height.

#### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
 })
 .Height("100%").Width("100%").ChildMapping("Children").TreeColumnIndex(1)
 .Render())
```

```
)
```

## RESPONSIVE.CS

```
public IActionResult Index()
{
 var tree = TreeGridItems.GetTreeData();
 ViewBag.datasource = tree;
 return View();
}
```

### Scroll to selected row

You can scroll the treegrid content to the selected row position by using the [RowSelected](#) event.

## CSHTML

```
@Html.EJS().NumericTextBox("numeric").Format("##").Min(0).Width(200).ShowSpinButton(false).Placeholder("Enter index to select a row").Change("onChange").Render()
@(Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 })
 .Height(270).Width("100%").ChildMapping("Children").TreeColumnIndex(1).RowSelected("rowSelected").Render()
)
<script>
 function onChange(args) {
 var treegrid = document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.selectRow(parseInt(this.getText(), 10));
 }
 function rowSelected(args) {
 var rowHeight =
 this.getRows()[treegrid.getSelectedRowIndex()[0]].scrollHeight;
 this.getContent().children[0].scrollTop = rowHeight *
 this.getSelectedRowIndex()[0];
 }
</script>
```

## SELECTED-ROW.CS

```
public IActionResult Index()
{
 var tree = TreeGridItems.GetTreeData();
 ViewBag.datasource = tree;
}
```

```
return View();
}
```

## Frozen rows and columns

### Frozen rows and columns

Frozen rows and columns provides an option to make rows and columns always visible in the top and left side of the tree grid while scrolling.

In this demo, the [FrozenColumns](#) is set as **2** and the [FrozenRows](#) is set as **3**. Hence, the left two columns and top three rows are frozen.

### CSHTML

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)ViewBag.datasource).AllowSelection(false).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task ID").Width(100).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(230).Add();
 col.Field("StartDate").HeaderText("Start Date").Width(150).TextAlign(TextAlign.Right).Format("yMd").Add();
 col.Field("EndDate").HeaderText("End Date").Width(150).TextAlign(TextAlign.Right).Format("yMd").Add();

 col.Field("Duration").HeaderText("Duration").Width(120).TextAlign(TextAlign.Right).Add();

 col.Field("Progress").HeaderText("Progress").Width(120).TextAlign(TextAlign.Right).Add();
 col.Field("Priority").HeaderText("Priority").Width(120).Add();

 col.Field("Approved").HeaderText("Approved").TextAlign(TextAlign.Left).Type("boolean").Width(110).Add();
}).Height(410).ChildMapping("Children").TreeColumnIndex(1).FrozenColumns(2).FrozenRows(3).Render()
```

### FREEZECOLUMN.CS

```
public IActionResult FrozenColumn()
{
 ViewBag.datasource = TreeData.GetDefaultData();
 return View();
}
```

### Freeze particular columns

To freeze particular column in the tree grid, the [IsFrozen](#) property can be used.

In this demo, the columns with field name **TaskName** and **StartDate** is frozen using the [IsFrozen](#) property.

### CSHTML

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)ViewBag.datasource).AllowSelection(false).Columns(col =>
```

```

{
 col.Field("TaskId").HeaderText("Task
ID").Width(100).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").Width(230).IsFrozen(true).Add();
 col.Field("StartDate").HeaderText("Start
Date").Width(150).IsFrozen(true).TextAlign(TextAlign.Right).Format("yMd").Ad
d();
 col.Field("EndDate").HeaderText("End
Date").Width(150).TextAlign(TextAlign.Right).Format("yMd").Add();

col.Field("Duration").HeaderText("Duration").Width(120).TextAlign(TextAlign.
Right).Add();

col.Field("Progress").HeaderText("Progress").Width(120).TextAlign(TextAlign.
Right).Add();
 col.Field("Priority").HeaderText("Priority").Width(120).Add();

col.Field("Approved").HeaderText("Approved").TextAlign(TextAlign.Left).Type(
"boolean").Width(110).Add();
 }).Height(410).ChildMapping("Children").Render()

```

### ISFREEZE.CS

```

public IActionResult FrozenColumn()
{
 ViewBag.datasource = TreeData.GetDefaultData();
 return View();
}

```

### Freeze direction

You can freeze the tree grid columns on the left or right side by using the [column.freeze](#) property and the remaining columns will be movable. The tree grid will automatically move the columns to the left or right position based on the [column.freeze](#) value.

Types of the [column.freeze](#) directions:

- **Left:** Allows you to freeze the columns at the left.
- **Right:** Allows you to freeze the columns at the right.

In this demo, the **Task Name** column is frozen at the left and the **Priority** column is frozen at the right side of the content table.

### CSHTML

```

@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<objec
t>)ViewBag.datasource).AllowSelection(false).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(100).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").Width(230).Freeze(Syncfusion.EJ2.Grids.FreezeDirection.Left).Add();
 col.Field("StartDate").HeaderText("Start
Date").Width(150).TextAlign(TextAlign.Right).Format("yMd").Add();

```

```

 col.Field("EndDate").HeaderText("End
Date").Width(150).TextAlign(TextAlign.Right).Format("yMd").Add();

col.Field("Duration").HeaderText("Duration").Width(120).TextAlign(TextAlign.
Right).Add();

col.Field("Progress").HeaderText("Progress").Width(120).TextAlign(TextAlign.
Right).Add();

col.Field("Priority").HeaderText("Priority").Width(120).Freeze(Syncfusion.EJ
2.Grids.FreezeDirection.Right).Add();

col.Field("Approved").HeaderText("Approved").TextAlign(TextAlign.Left).Type(
"boolean").Width(110).Add();
 }).Height(410).TreeColumnIndex(1).ChildMapping("Children").Render()

```

### **FREEZE-DIRECTION.CS**

```

public IActionResult FrozenColumn()
{
 ViewBag.datasource = TreeData.GetDefaultData();
 return View();
}

```

#### *Limitations of frozen tree grid*

The following features are not supported in frozen rows and columns:

- Row Template
- Detail Template
- Cell Editing

Freeze Direction feature has the below limitations, along with the above mentioned limitations.

- Infinite scroll cache mode
- Freeze direction in the stacked header is not compatible with column reordering.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Tree Grid Virtualization

TreeGrid allows you to load large amount of data without performance degradation.

#### Row Virtualization

Row virtualization allows you to load and render rows only in the content viewport. It is an alternative way of paging in which the rows will be appended while scrolling vertically. To setup the row virtualization, you need to define [EnableVirtualization](#) as true and content height by [Height](#) property.

The number of records displayed in the TreeGrid is determined implicitly by height of the content area and a buffer records will be maintained in the TreeGrid content in addition to the original set of rows.

Expand and Collapse state of any child record will be persisted.

**CSHTML**

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskID").HeaderText("Player Jersey").Width(120).TextAlign(TextAlign.Right).Add();
 col.Field("FIELD1").HeaderText("Player Name").Width(120).Add();

 col.Field("FIELD2").HeaderText("Year").Width(100).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD3").HeaderText("Stint").Width(120).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD4").HeaderText("TMID").Width(120).TextAlign(TextAlign.Right).Add();

}).Height(400).ChildMapping("Children").TreeColumnIndex(1).EnableVirtualization(true).Render()
```

**ROWVIRTUAL.CS**

```
public IActionResult VirtualScrolling()
{
 ViewBag.datasource = VirtualDataFormat.GetVirtualData();
 return View();
}
```

**Column Virtualization**

Column virtualization allows you to virtualize columns. It will render column only in the current view port and all other columns are rendered on demand during horizontal scrolling.

To setup the column virtualization, set the [EnableVirtualization](#) and [EnableColumnVirtualization](#) properties as **true**.

**CSHTML**

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskID").HeaderText("Player Jersey").Width(120).TextAlign(TextAlign.Right).Add();
 col.Field("FIELD1").HeaderText("Player Name").Width(120).Add();

 col.Field("FIELD2").HeaderText("Year").Width(100).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD3").HeaderText("Stint").Width(120).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD4").HeaderText("TMID").Width(120).TextAlign(TextAlign.Right).Add();

 col.Field("Field6").HeaderText("GP").Width("120").TextAlign(TextAlign.Right).Add();
}
```



```
col.Field("Field7").HeaderText("GS").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field8").HeaderText("Minutes").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field9").HeaderText("Points").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field10").HeaderText("oRebounds").Width("130").TextAlign(TextAlign.Right)
.Add();

col.Field("Field11").HeaderText("dRebounds").Width("130").TextAlign(TextAlign.Right)
.Add();

col.Field("Field12").HeaderText("Rebounds").Width("130").TextAlign(TextAlign.Right)
.Add();

col.Field("Field13").HeaderText("Assists").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field14").HeaderText("Steals").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field15").HeaderText("Blocks").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field16").HeaderText("Turnovers").Width("130").TextAlign(TextAlign.Right)
.Add();

col.Field("Field17").HeaderText("PF").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field18").HeaderText("fgAttempted").Width("150").TextAlign(TextAlign.Right)
.Add();

col.Field("Field19").HeaderText("fgMade").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field20").HeaderText("ftAttempted").Width("150").TextAlign(TextAlign.Right)
.Add();

col.Field("Field21").HeaderText("ftMade").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field22").HeaderText("ThreeAttempted").Width("150").TextAlign(TextAlign.Right)
.Add();

col.Field("Field23").HeaderText("ThreeMade").Width("130").TextAlign(TextAlign.Right)
.Add();

col.Field("Field24").HeaderText("PostGP").Width("120").TextAlign(TextAlign.Right)
.Add();

col.Field("Field25").HeaderText("PostGS").Width("120").TextAlign(TextAlign.Right)
.Add();
```

```
col.Field("Field26").HeaderText("PostMinutes").Width("120").TextAlign(TextAlign.Right).Add();

col.Field("Field27").HeaderText("PostPoints").Width("130").TextAlign(TextAlign.Right).Add();

col.Field("Field28").HeaderText("PostoRebounds").Width("130").TextAlign(TextAlign.Right).Add();

col.Field("Field29").HeaderText("PostdRebounds").Width("130").TextAlign(TextAlign.Right).Add();

col.Field("Field30").HeaderText("PostRebounds").Width("130").TextAlign(TextAlign.Right).Add();

}).Height(400).ChildMapping("Children").TreeColumnIndex(1).EnableVirtualization(true).EnableColumnVirtualization(true).Render()
```

### **COLUMNVIRTUAL.CS**

```
public IActionResult VirtualScrolling()
{
 ViewBag.datasource = VirtualDataFormat.GetVirtualData();
 return View();
}
```

**Note:** Column's **Width** is required for column virtualization. If column's width is not defined then tree grid will consider its value as **200px**.

### Limitations for Virtualization

- Due to the element height limitation in browsers, the maximum number of records loaded by the treegrid is limited by the browser capability.
- Cell selection will not be persisted in row.
- Virtual scrolling is not compatible with detail template, clipboard functionality and row drag and drop features.
- The page size provided must be two times larger than the number of visible rows in the TreeGrid. If the page size is failed to meet this condition then the size will be determined by TreeGrid.
- The virtual height of the treegrid content is calculated using the row height and total number of records in the data source and hence features which changes row height such as text wrapping are not supported. If you want to increase the row height to accommodate the content then you can specify the row height as below to ensure all the table rows are in same height.

`css

```
.e-treegrid .e-row {
height: 2em;
}
```

- Programmatic selection using the **SelectRows** method is not supported in virtual scrolling.
- When virtualization is active in a tree grid, the editCell method is unusable for records outside the currently visible viewport.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Tree Grid Infinite scrolling

Infinite scrolling is used to load a huge amount of data without degrading the Tree Grid performance. This feature works like the lazy loading concept, which means the buffer data is loaded only when the scrollbar reaches the end of the scroller.

To use Infinite scrolling, set **EnableInfiniteScrolling** property as true.

**Note:** \* In this feature, Tree Grid will not make a new data request when you visit the same page again.

### CSHTML

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)
ViewBag.datasource).Columns(col =>
{
 col.Field("TaskID").HeaderText("Player
Jersey").Width(120).TextAlign(TextAlign.Right).Add();
 col.Field("FIELD1").HeaderText("Player Name").Width(120).Add();

 col.Field("FIELD2").HeaderText("Year").Width(100).TextAlign(TextAlign.Right)
.Add();

 col.Field("FIELD3").HeaderText("Stint").Width(120).TextAlign(TextAlign.Right)
.Add();

 col.Field("FIELD4").HeaderText("TMID").Width(120).TextAlign(TextAlign.Right)
.Add();

}).Height(400).ChildMapping("Children").TreeColumnIndex(1).EnableInfiniteSc
rolling().Render()
```

### INFINITESROLL.CS

```
public IActionResult InfiniteScrolling()
{
 ViewBag.datasource = VirtualDataFormat.GetVirtualData();
 return View();
}
```

### InitialBlocks

You can define the initial loading pages count by using **InitialBlocks** property of **InfiniteScrollSettings**. By default, this feature loads three pages in initial rendering.

In the below demo, we have changed this property value to load five page records instead of three.

**CSHTML**

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskID").HeaderText("Player Jersey").Width(120).TextAlign(TextAlign.Right).Add();
 col.Field("FIELD1").HeaderText("Player Name").Width(120).Add();

 col.Field("FIELD2").HeaderText("Year").Width(100).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD3").HeaderText("Stint").Width(120).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD4").HeaderText("TMID").Width(120).TextAlign(TextAlign.Right).Add();

}).Height(400).ChildMapping("Children").TreeColumnIndex(1).EnableInfiniteScrolling().InfiniteScrollSettings(settings =>
{settings.InitialBlocks(5);}).Render()
```

**INFINITESCROLL.CS**

```
public IActionResult InfiniteScrolling()
{
 ViewBag.datasource = VirtualDataFormat.GetVirtualData();
 return View();
}
```

**Cache Mode**

Cache is used to store the loaded rows object in the Tree Grid instance which can be reused for creating the row elements whenever you scroll to already visited page. Also, this mode maintains row elements based on the **MaxBlocks** count value of **InfiniteScrollSettings**, once this limit exceeds then it will remove row elements from DOM for new rows.

To enable the cache mode in Infinite scrolling, set **EnableCache** property of **InfiniteScrollSettings** as true.

**CSHTML**

```
@Html.EJS().TreeGrid("DefaultFunctionalities").DataSource((IEnumerable<object>)ViewBag.datasource).Columns(col =>
{
 col.Field("TaskID").HeaderText("Player Jersey").Width(120).TextAlign(TextAlign.Right).Add();
 col.Field("FIELD1").HeaderText("Player Name").Width(120).Add();

 col.Field("FIELD2").HeaderText("Year").Width(100).TextAlign(TextAlign.Right).Add();

 col.Field("FIELD3").HeaderText("Stint").Width(120).TextAlign(TextAlign.Right).Add();

}
```

```
col.Field("FIELD4").HeaderText("TMID").Width(120).TextAlign(TextAlign.Right)
.Add();

}).Height(400).ChildMapping("Children").TreeColumnIndex(1).EnableInfiniteSc
rolling(true).InfiniteScrollSettings(settings =>
{settings.EnableCache(true);}).Render()
```

### INFINITESCROLL.CS

```
public IActionResult InfiniteScrolling()
{
 ViewBag.datasource = VirtualDataFormat.GetVirtualData();
 return View();
}
```

### Limitations for Infinite Scrolling

- Due to the element height limitation in browsers, the maximum number of records loaded by the tree grid is limited due to the browser capability.
- Initial loading rows total height must be greater than the viewport height.
- Cell selection will not be persisted in cache mode.
- Infinite scrolling is not compatible with batch editing, cell editing, detail template and hierarchy features.
- The aggregated information and total group items are displayed based on the current view items. To get these information regardless of the view items, refer to the
- Programmatic selection using the [selectRows](#) and [selectRow](#) method is not supported in infinite scrolling.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### State Persistence

State persistence refers to the TreeGrid's state maintained in the browser's [localStorage](#) even if the browser is refreshed or if you move to the next page within the browser.

State persistence stores treegrid's model object in the local storage when the [enablePersistence](#) is defined as true.

#### Get or set localStorage value

If the [enablePersistence](#) property is set to true, the treegrid property value is saved in the **window.localStorage** for reference. You can get/set the localStorage value by using the getItem/setItem method in the **window.localStorage**.

```
`typescript
```

```
//get the TreeGrid model.
```

```
var value = window.localStorage.getItem('treegridTreeGrid'); //"treegridTreeGrid" is component name +
component id.
```

```
var model = JSON.parse(model);
`typescript
//set the TreeGrid model.

window.localStorage.setItem('treegridTreeGrid', JSON.stringify(model)); //"treegridTreeGrid" is
component name + component id.
```

### Print

To print the TreeGrid, use the [print](#) method from treegrid instance. The print option can be displayed on the [Toolbar](#) by adding the **print** toolbar item.

### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>) ViewBag.data
source).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
}).Height(265).ChildMapping("Children").TreeColumnIndex(1).Toolbar(new
List<string>() { "Print" }).Render())
```

### PRINT.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.dataSource = tree;
 return View();
}
```

### Page setup

Some of the print options cannot be configured through JavaScript code. So, you have to customize the layout, paper size, and margin options using the browser page setup dialog. Refer to the following links to know more about the browser page setup:

- [Chrome](#)
- [Firefox](#)
- [Safari](#)
- [IE](#)

### Print using an external button

To print the treegrid from an external button, invoke the [print](#) method.

#### CSHTML

```
@Html.EJS().Button("print").Content("Print").Render()
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 })
 .Height(265)
 .ChildMapping("Children")
 .TreeColumnIndex(1)
 .Render())
<script>
 document.getElementById('print').onclick = function () {
 var treegrid = document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.print();
 }
</script>
```

#### PRINT-BUTTON.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

### Print the visible page

By default, the treegrid prints all the pages. To print the current page alone, set the [PrintMode](#) to **CurrentPage**.

#### CSHTML

```
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 })
 .Height(265)
 .ChildMapping("Children")
 .TreeColumnIndex(1)
 .Render())
```

```

 }).Height(220).AllowPaging().PageSettings(p =>
 p.PageSize(8)).ChildMapping("Children").TreeColumnIndex(1)
 .Toolbar(new List<string>() { "Print"
 }).PrintMode(Syncfusion.EJ2.Grids.PrintMode.CurrentPage).Render()
)

```

### CURRENT-PAGE.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Print large number of columns

By default, the browser uses A4 as page size option to print pages and to adapt the size of the page the browser print preview will auto-hide the overflowed contents. Hence treegrid with large number of columns will cut off to adapt the print page.

To show large number of columns when printing, adjust the scale option from print option panel based on your content size.



### Show or Hide columns while Printing

You can show a hidden column or hide a visible column while printing the treegrid using [ToolbarClick](#) and [PrintComplete](#) events.

In [ToolbarClick](#) event, based on **args.item.text** as **print**. We can show or hide columns by setting [Visible](#) of [Column](#) property to **true** or **false** respectively.

In [PrintComplete](#) event, We have reversed the state back to the previous state.

In the below example, we have **Duration** as a hidden column in the treegrid. While printing, we have changed **Duration** to visible column and **StartDate** as hidden column.



**CSHTML**

```

@(Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>) ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.R
ight).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Visible(false).Add();
 }).Height(265).ChildMapping("Children").TreeColumnIndex(1).Toolbar(new
List<string>() { "Print" })

 .ToolbarClick("toolbarClick").PrintComplete("printComplete").Render()
)
<script>
 function toolbarClick(args) {
 if (args.item.text === 'Print') {
 var cols = this.grid.columns;
 for (var i = 0; i < cols.length; i++) {
 if (cols[i].field == "Duration") {
 cols[i].visible = true;
 }
 else if (cols[i].field == "StartDate") {
 cols[i].visible = false;
 }
 }
 }
 }
 function printComplete(args) {
 var cols = this.grid.columns;
 for (var i = 0; i < cols.length; i++) {
 if (cols[i].field == "Duration") {
 cols[i].visible = false;
 }
 else if (cols[i].field == "StartDate") {
 cols[i].visible = true;
 }
 }
 }
}
</script>

```

**SHOW-HIDE.CS**

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Limitations of Printing Large Data

When treegrid contains large number of data, printing all the data at once is not a best option for the browser performance. Because to render all the DOM elements in one page will produce performance issues in the browser. It leads to browser slow down or browser hang.

If printing of all the data is still needed, we suggest to Export the treegrid to **Excel** or **CSV** or **Pdf** and then print it from another non-web based application.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Adaptive View in ASP.NET MVC Tree Grid Component

The Tree Grid user interface (UI) was redesigned to provide an optimal viewing experience and improve usability on small screens. This interface will render the filter, sort, and edit dialogs adaptively.

#### Render adaptive dialogs

When you enable the [enableAdaptiveUI](#) property, the tree grid will render the filter, sort, and edit dialogs in full screen for a better user experience. The following demo demonstrates this behavior.

#### CSHTML

```
@Html.EJS().TreeGrid("adaptivebrowser").DataSource((IEnumerable<object>)View
Bag.datasource).TreeColumnIndex(1).Height("100%").EnableAdaptiveUI(true).All
owPaging(true).AllowFiltering(true).ChildMapping("Children").AllowSorting(tr
ue).Columns(col =>
{
 col.Field("TaskId").HeaderText("Task
ID").Width(135).IsPrimaryKey(true).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(280).Add();

col.Field("Duration").HeaderText("Duration").Width(140).TextAlign(TextAlign.
Right).Add();

col.Field("Progress").HeaderText("Progress").Width(145).TextAlign(TextAlign.
Right).Add();
 }).FilterSettings(filter => {
filter.Type(Syncfusion.EJ2.TreeGrid.FilterType.Excel); }).EditSettings(edit
=>
{
edit.AllowAdding(true).AllowEditing(true).AllowDeleting(true).Mode(Syncfusio
n.EJ2.TreeGrid.EditMode.Dialog);
 }).Toolbar(new List<string>() { "Add", "Edit", "Delete", "Update",
"Cancel", "Search" }).Load("load").Render()
```

#### DEFAULT.CS

```
public IActionResult Adaptive()
{
 var order = TreeData.GetDefaultData();
 ViewBag.dataSource = order;
 return View();
}
```

**Note:** Refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to learn how to present and manipulate data.

### Loading Animation in ASP.NET MVC Tree Grid Component

The Tree Grid displays a loading indicator while the data is being fetched and bound to the tree grid during initial rendering, refreshing, and after performing any tree grid actions like sorting, filtering, and more.

The tree grid supports two indicator types, which can be enabled by setting the `loadingIndicator.indicatorType` property to `Spinner` or `Shimmer`. The default value of the indicator type is `Spinner`.

In the following sample, the Shimmer indicator is displayed while the tree grid is loading and refreshing when using the remote data.

#### CSHTML

```
@Html.EJS().TreeGrid("RemoteData").DataSource(dataManager =>
dataManager.Url("https://ej2services.syncfusion.com/production/web-
services/api/SelfReferenceData").Adaptor("WebApiAdaptor").CrossDomain(true))
.Columns(col =>
{
 col.Field("TaskID").HeaderText("Task
ID").Width(120).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").Width(240).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Left).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(1
40).Add();

 col.Field("Duration").HeaderText("Duration").Width(130).TextAlign(Syncfusion
.EJ2.Grids.TextAlign.Right).Add();
 col.Field("Progress").HeaderText("Progress").Width(130).Add();
}).HasChildMapping("isParent").Height(400).IdMapping("TaskID").ParentIdMapping
("ParentItem").AllowPaging().AllowSorting().TreeColumnIndex(1).PageSetting
s(page=>page.PageCount(3)).LoadingIndicator(l => {
 l.IndicatorType(Syncfusion.EJ2.Grids.IndicatorType.Shimmer); }).Render()
```

#### LOADINGANIMATION.CS

```
public ActionResult LoadingAnimation()
{
 return View();
}
```

You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### PDF Export in ASP.NET MVC Tree Grid Component

PDF export allows exporting TreeGrid data to PDF document. You need to use the [pdfExport](#) method for exporting. To enable PDF export in the treegrid, set the [AllowPdfExport](#) as true.

**CSHTML**

```
@(Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").Width(90).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start Date").Format("yMd").Type("date").TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Width(120).Add();

 col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(Syncfusion.EJ2.Grids.TextAlign.Right).Add();

 })
 .Height(220)
 .ChildMapping("Children")
 .TreeColumnIndex(1)
 .Toolbar(new List<string>() { "PdfExport" })
 .AllowPdfExport()
 .AllowPaging()
 .PageSettings(page => page.PageSize(7))
 .ToolbarClick("toolbarClick")
 .Render()
)
<script>
 function toolbarClick(args) {
 if (args['item'].text === 'PDF Export') {
 var treegrid =
 document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.pdfExport();
 }
 }
</script>
```

**EXPORT.CS**

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

**Excel Export in ASP.NET MVC Tree Grid Component**

The excel export allows exporting TreeGrid data to Excel document. You need to use the [excelExport](#) method for exporting. To enable Excel export in the treegrid, set the [AllowExcelExport](#) as true.

**CSHTML**

```
@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.datasource)
 .Columns(col =>
```

```

 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(TextAlign.Right).Width(120).Add(
);

col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(TextAlign.
Right).Add();

}).Height(220).ChildMapping("Children").TreeColumnIndex(1).Toolbar(new
List<string>() { "ExcelExport" })
 .AllowExcelExport().AllowPaging().PageSettings(page =>
page.PageSize(7)).ToolbarClick("toolbarClick")
 .Render()
)
<script>
 function toolbarClick(args) {
 if (args['item'].text === 'Excel Export') {
 var treegrid =
document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.excelExport();
 }
 }
</script>

```

### EXPORT.CS

```

public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}

```

### Persist collapsed state

You can persist the collapsed state in the exported document by defining `isCollapsedStatePersist` property as true in `TreeGridExcelExportProperties` parameter of `excelExport` method.

### CSHTML

```

@using Syncfusion.EJ2.Grids
@ (Html.EJS().TreeGrid("TreeGrid").DataSource((IEnumerable<object>)ViewBag.da
tasource)
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(90).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(180).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").Type("date").TextAlign(TextAlign.Right).Width(120).Add(
);

```

```
col.Field("Duration").HeaderText("Duration").Width(110).TextAlign(TextAlign.Right).Add();

}).Height(220).ChildMapping("Children").TreeColumnIndex(1).Toolbar(new
List<string>() { "ExcelExport" })
 .AllowExcelExport().AllowPaging().PageSettings(page =>
page.PageSize(7)).ToolbarClick("toolbarClick")
 .Render()
)
<script>
 function toolbarClick(args) {
 if (args['item'].text === 'Excel Export') {
 var excelExportProperties = {
 isCollapsedStatePersist: true
 };
 var treegrid =
document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.excelExport(excelExportProperties);
 }
 }
</script>
```

### IS-COLLAPSED.CS

```
public IActionResult Index()
{
 var tree = TreeData.GetDefaultData();
 ViewBag.datasource = tree;
 return View();
}
```

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

### Clipboard

The clipboard provides an option to copy selected rows or cells data into the clipboard.

The following list of keyboard shortcuts is supported in the Tree Grid to copy selected rows or cells data into the clipboard.

Interaction keys | Description

**Ctrl + C | Copy selected rows or cells data into clipboard.**

**Ctrl + Shift + H | Copy selected rows or cells data with header into clipboard.**

### CSHTML

```
@(Html.EJS().TreeGrid("Clipboard")
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .SelectionSettings(select =>
 {
 select.Mode(SelectionMode.Row);
 select.Type(SelectionType.Multiple);
 })
)
```

```

 })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();
 }).ChildMapping("Children").TreeColumnIndex(1).Height(200).Render()

```

### **COPY.CS**

```

public IActionResult Index()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}

```

### Copy to clipboard by external buttons

To copy selected rows or cells data into the clipboard with help of external buttons, you need to invoke the [copy](#) method.

### **CSHTML**

```

@Html.EJS().Button("Copy").Content("Copy").IsPrimary(true).Render()
@Html.EJS().Button("CopyHeader").Content("Copy With
Header").IsPrimary(true).Render()
@(Html.EJS().TreeGrid("CopyBtn")
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .SelectionSettings(select =>
 {
 select.Mode(SelectionMode.Row);
 select.Type(SelectionType.Multiple);
 })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(110).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();
 }).ChildMapping("Children").TreeColumnIndex(1).Height(200).Render())
<script>
 document.getElementById("Copy").onclick = function () {
 var treegrid = document.getElementById("CopyBtn").ej2_instances[0];
 treegrid.copy();
 }

```

```

 }
 document.getElementById("CopyHeader").onclick = function () {
 var treegrid = document.getElementById("CopyBtn").ej2_instances[0];
 treegrid.copy(true);
 }
</script>

```

### COPYBUTTONS.CS

```

public IActionResult Index()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}

```

### Copy Hierarchy Modes

Tree Grid provides support for a set of copy modes with **CopyHierarchyMode** property.

The below are the type of copy mode available in TreeGrid.

- **Parent** : This is the default copy hierarchy mode in Tree Grid. Clipboard value have the selected records with its parent records, if the selected records not have any parent record then the selected record will be in clipboard.
- **Child** : Clipboard value will have the selected records with its child record. If the selected records do not have any child record then the selected records will be in clipboard.
- **Both** : Clipboard value will have the selected records with its both parent and child record. If the selected records do not have any parent and child record then the selected records alone in clipboard.
- **None** : Only the Selected records will be in clipboard.

### CSHTML

```

@{
 List<object> toolbar = new List<object>();
 toolbar.Add(new { text = "Copy", tooltipText = "Copy", prefixIcon = "e-copy", id = "copy" });
 toolbar.Add(new { text = "Copy With Header", tooltipText = "Copy With Header", prefixIcon = "e-copy", id = "copyHeader" });
}
@Html.EJS().DropDownList("SearchMode").Width("50%").DataSource((IEnumerable<object>)ViewBag.dropdata).Value("Parent").Fields(new Syncfusion.EJ2.DropDowns.DropDownListFieldSettings { Text = "id", Value = "mode" }).Change("onChange").Render()
@ (Html.EJS().TreeGrid("TreeGrid")
 .DataSource((IEnumerable<object>)ViewBag.datasource)
 .AllowPaging()
 .Toolbar(toolbar)
 .ToolbarClick("toolbarClick")
 .SelectionSettings(sel =>
 {
 sel.Type(SelectionType.Multiple);
 })
)

```



```

 .PageSettings(p => p.PageSize(10))
 .AllowSelection()
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task
ID").Width(80).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task
Name").Width(200).Add();
 col.Field("StartDate").HeaderText("Start
Date").Format("yMd").TextAlign(TextAlign.Right).Width(100).Add();

col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(90).Add();

col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();
 })
 .ChildMapping("Children").TreeColumnIndex(1).Render()
<script>
function toolbarClick(args) {
 if (this.getSelectedRecords().length > 0) {
 var withHeader = false;
 if (args.item.id === 'copyHeader') {
 withHeader = true;
 }
 this.copy(withHeader);
 } else {
 var dialogObj =
document.getElementById('alert_dialog').ej2_instances[0];
 dialogObj.show();
 }
}
function onChange(e) {
 var mode = e.value;
 var treegrid =
document.getElementById("TreeGrid").ej2_instances[0];
 treegrid.copyHierarchyMode = mode;
}
</script>

```

## HIERARCHY.CS

```

public IActionResult Index()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 List<Object> dropData = new List<object>() {
 new { id = "Parent", mode = "Parent" },
 new { id = "Child", mode = "Child" },
 new { id = "Both", mode = "Both" },
 new { id = "None", mode = "None" }
 };
 ViewBag.dropdata = dropData;
 return View();
}

```

## AutoFill

AutoFill Feature allows you to copy the data of selected cells and paste it to another cells by just dragging the autofill icon of the selected cells up to required cells. This feature is enabled by defining `EnableAutoFill` property as true.

### CSHTML

```
@(Html.EJS().TreeGrid("AutoFill")
 .DataSource((IEnumerable<object>) ViewBag.dataSource)
 .SelectionSettings(select =>
 {
 select.Mode(SelectionMode.Cell);
 select.CellSelectionMode(CellSelectionMode.Box);
 select.Type(SelectionType.Multiple);
 })
 .Toolbar(new List<string>() { "Add", "Update", "Cancel" })
 .EditSettings(edit =>
 {
 edit.AllowAdding(true);
 edit.AllowEditing(true);
 edit.AllowDeleting(true);
 edit.Mode(Syncfusion.EJ2.TreeGrid.EditMode.Batch);
 })
 .Columns(col =>
 {
 col.Field("TaskId").HeaderText("Task ID").IsPrimaryKey(true).Width(110).TextAlign(TextAlign.Right).Add();
 col.Field("TaskName").HeaderText("Task Name").Width(210).Add();

 col.Field("Duration").HeaderText("Duration").TextAlign(TextAlign.Right).Width(110).Add();

 col.Field("Progress").HeaderText("Progress").TextAlign(TextAlign.Right).Width(90).Add();

 }).ChildMapping("Children").TreeColumnIndex(1).Height(200).EnableAutoFill(true).Render())
```

### AUTOFILL.CS

```
public IActionResult Index()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

**Note:** \* If `EnableAutoFill` is set to true, then the autofill icon will be displayed on cell selection to copy cells.

\* It requires the selection `Mode` to be `Cell`, `CellSelectionMode` to be `Box` and also Batch Editing should be enabled.

*Limitations of AutoFill*

- Since the string values are not parsed to number and date type, so when the selected string type cells are dragged to number type cells then it will display as **NaN**. For date type cells, when the selected string type cells are dragged to date type cells then it will display as an **empty cell**.
- Linear series and the sequential data generations are not supported in this autofill feature.

*Paste*

You can able to copy the content of a cell or a group of cells by selecting the cells and pressing Ctrl + C shortcut key and paste it to another set of cells by selecting the cells and pressing Ctrl + V shortcut key.

**CSHTML**

```
@ (Html.EJS () .TreeGrid ("Paste")
 .DataSource ((IEnumerable<object>) ViewBag.dataSource)
 .SelectionSettings (select =>
 {
 select.Mode (SelectionMode.Cell);
 select.CellSelectionMode (CellSelectionMode.Box);
 select.Type (SelectionType.Multiple);
 })
 .Toolbar (new List<string> () { "Add", "Update", "Cancel" })
 .EditSettings (edit =>
 {
 edit.AllowAdding (true);
 edit.AllowEditing (true);
 edit.AllowDeleting (true);
 edit.Mode (Syncfusion.EJ2.TreeGrid.EditMode.Batch);
 })
 .Columns (col =>
 {
 col.Field ("TaskId").HeaderText ("Task
ID").IsPrimaryKey (true).Width (110).TextAlign (TextAlign.Right).Add ();
 col.Field ("TaskName").HeaderText ("Task Name").Width (210).Add ();

 col.Field ("Duration").HeaderText ("Duration").TextAlign (TextAlign.Right).Width
(110).Add ();

 col.Field ("Progress").HeaderText ("Progress").TextAlign (TextAlign.Right).Width
(90).Add ();
 }) .ChildMapping ("Children").TreeColumnIndex (1).Height (200).Render ())
```

**PASTE.CS**

```
public IActionResult Index()
{
 var treeData = TreeGridItems.GetTreeData();
 ViewBag.datasource = treeData;
 return View();
}
```

**Note:** To perform paste functionality, it requires the selection **Mode** to be **Cell**, **CellSelectionMode** to be **Box** and also Batch Editing should be enabled.

#### *Limitations of Paste Functionality*

- Since the string values are not parsed to number and date type, so when the copied string type cells are pasted to number type cells then it will display as **NaN**. For date type cells, when the copied string format cells are pasted to date type cells then it will display as an **empty cell**.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

#### Accessibility in Tree Grid control

The Tree Grid component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Tree Grid component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

| Right-To-Left Support |  |

| Color Contrast |  |

| Mobile Device Support |  |

| Keyboard Navigation Support |  |

| [Accessibility Checker](#) Validation |  |

| [Axe-core](#) Accessibility Validation |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

```
}
</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>
```

### WAI-ARIA attributes

The Tree Grid component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the Tree Grid component:

| Attributes                 | Purpose                                                                                                                                                                 |
|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ---                        | ---                                                                                                                                                                     |
| <code>role=treegrid</code> | Used to convey a significant and contextual message to the user.                                                                                                        |
| <code>aria-selected</code> | Accurately reflect the selection state, whether it's single-select or multi-select.                                                                                     |
| <code>aria-expanded</code> | It can be used to show whether a node is expanded or collapsed, making it easier for screen reader users to navigate and understand the hierarchy.                      |
| <code>aria-sort</code>     | Indicate the current sorting order of a table column for users with disabilities, facilitating accessible data presentation and interaction.                            |
| <code>aria-busy</code>     | Loading state to improve accessibility for users, particularly those relying on screen readers.                                                                         |
| <code>aria-invalid</code>  | To indicate whether the user's input in a form field is valid or invalid, aiding users, including those with disabilities, in understanding and correcting their input. |
| <code>aria-grabbed</code>  | Provides accessibility information for users interacting with draggable elements                                                                                        |
| <code>aria-owns</code>     | Establishing relationships between an element and the elements it owns or controls.                                                                                     |
| <code>aria-label</code>    | Provides an accessible name for the close icon.                                                                                                                         |

### Keyboard interaction

The Tree Grid component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the Tree Grid component.

Interaction Keys | Description

**PageDown** | Goes to the next page.

**PageUp** | Goes to the previous page.

**Ctrl + Alt +PageDown** | Goes to the last page.

**Ctrl + Alt + PageUp** | Goes to the first page.

**Alt + PageDown** | Goes to the next page.

**Alt + PageUp** | Goes to the previous page.

**Home** | Goes to the first cell.

**End** | Goes to the last cell.

**Ctrl + Home** | Goes to the first row.

**Ctrl + End** | Goes to the last row.

**DownArrow** | Moves the cell focus downward.

**UpArrow** | Moves the cell focus upward.

**LeftArrow** | Moves the cell focus left side.

**RightArrow** | Moves the cell focus right side.

**Shift + DownArrow** | Extends the row/cell selection downwards.

**Shift + UpArrow** | Extends the row/cell selection upwards.

**Shift + LeftArrow** | Extends the cell selection to the left side.

**Shift + RightArrow** | Extends the cell selection to the right side.

**Enter** | Moves the row/cell selection downward. If current cell is in edit state, then completes the editing. If the current cell is a header then performs sorting.

**Shift + Enter** | Moves the row/cell selection upward. If the current cell is a header then clears sorting for the selected column.

**Ctrl + Enter** | If the current cell is a header then performs multi-sorting.

**Tab** | Moves the cell selection right side.

**Shift + Tab** | Moves the cell selection left side.

**Esc** | Deselects all the rows/cells.

**Ctrl + A** | Selects all the rows/cells.

**UpArrow** | Moves up a row/cell selection.

**DownArrow** | Moves down a row/cell selection.

**RightArrow** | Moves to the right cell selection.

**LeftArrow** | Moves to the left cell selection.

**Ctrl + Shift + DownArrow** | Expands the selected group.

**Ctrl + DownArrow** | Expands all the visible groups.

**Ctrl + Shift + UpArrow** | Collapses the selected group.

**Ctrl + UpArrow** | Collapses all the visible groups.

**Ctrl + P** | Prints the TreeGrid.

### Ensuring accessibility

The Tree Grid component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Tree Grid component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Tree Grid component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Styling

To modify the TreeGrid appearance, you need to override the default CSS of treegrid. Find the list of CSS classes and its corresponding section in treegrid. Also, you have an option to create your own custom theme for all the ASP.NET MVC controls using our [Theme Studio](#).

Section | CSS class | Purpose of CSS class

**Root** | e-treegrid | This classes are in this root element (div) of the treegrid control.

**Header** | e-gridheader | This class is added in the root element of header element. In this class, You can override thin line between header and content of the treegrid.

| e-table | This class is added at 'table' of the treegrid header. This CSS class makes table width as 100 %.

| e-columnheader | This class is added at 'tr' of the treegrid header.

| e-headercell | This class is added in 'th' element of treegrid header. You can override background color of header and border color.

| e-headercelldiv | This class is add in div which present 'th' element in the header. we recommend you to use the e-headercelldiv to override skeleton of header.

**Body** | e-gridcontent | This class is added at root of body content. This is to override background color of the body.

| e-table | This class is added to table of content. This CSS class makes table width as 100 %.

| e-altrow | This class is added to alternate rows of treegrid. This is to override alternate row color of the treegrid.

| e-rowcell | This class is added to all cells in the treegrid. This is to override cells appearance and styling.

| e-groupcaption | This class is added to the 'td' of group caption which is to change the background color of caption cell.

| e-selectionbackground | This class is added to rowcell's of the treegrid. This is override selection.

| e-hover | This class adds to row of treegrid, while hovering the treegrid rows.

**Pager** | e-pager | This class is added to root element of the pager. This to change appearance of the background color and color of font.

| e-pagercontainer | This class is added to numeric items of the pager.

| e-parentmsgbar | This class is added to pager info of the pager.

**Summary**|e-gridfooter| This class is added to root of the summary div.

| |e-summaryrow| This class is added to rows of treegrid summary.

| |e-summarycell| This class is added to cells of summary row. This to override background color of summary.

**Note:** You can refer to our [ASP.NET MVC Tree Grid](#) feature tour page for its groundbreaking feature representations. You can also explore our [ASP.NET MVC Tree Grid example](#) to know how to present and manipulate data.

## TreeMap

### Getting Started with ASP.NET MVC TreeMap Control

This section briefly explains about how to include [ASP.NET MVC TreeMap](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,



**Note:** Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Add script resources

Here, the script is referred using CDN inside the `<head>` of `~/Pages/Shared/_Layout.cshtml` file as follows,

##### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

##### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

#### Add ASP.NET MVC TreeMap control

Now, add the Syncfusion ASP.NET MVC TreeMap control in `~/Views/Home/Index.cshtml` page.

##### CSHTML

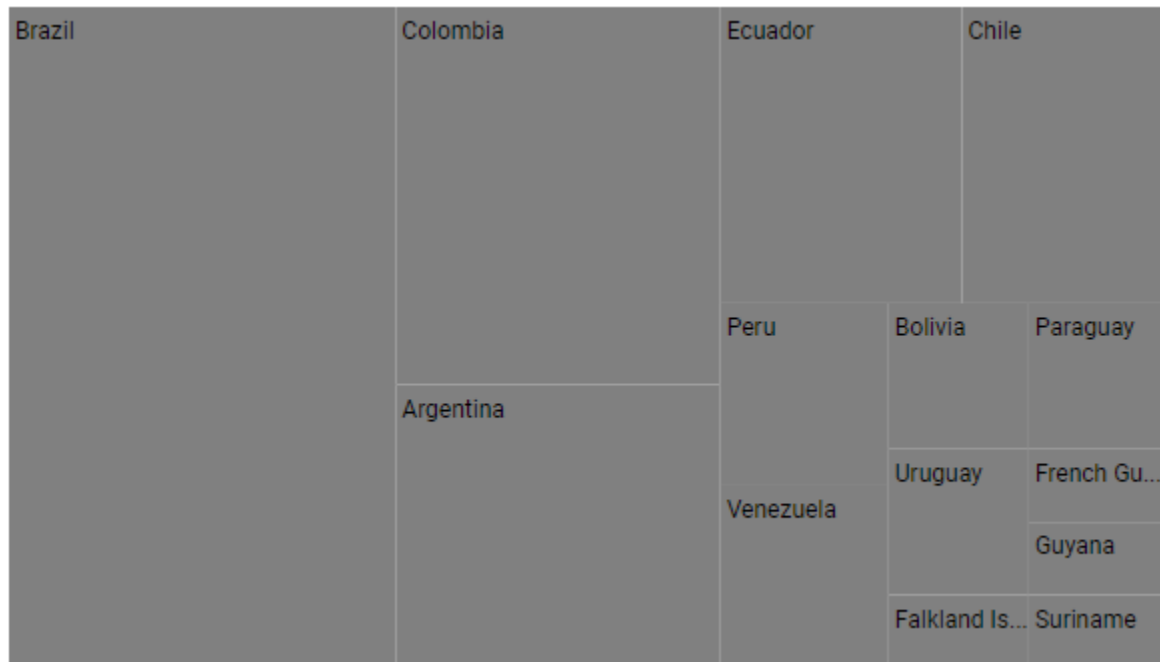
```
<h2> Essential JS 2 for ASP.NET MVC TreeMap </h2>
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("Count").LeafItemSettings(leaf=>leaf.LabelPath("State")).Render()
</div>
<script>
function load(args)
{
var data = [
{ Title: 'State wise International Airport count in South America', State: "Brazil", Count: 25 },
{ Title: 'State wise International Airport count in South America', State: "Colombia", Count: 12 },
{ Title: 'State wise International Airport count in South America', State: "Argentina", Count: 9 },
{ Title: 'State wise International Airport count in South America', State: "Ecuador", Count: 7 },
{ Title: 'State wise International Airport count in South America', State: "Chile", Count: 6 },
```

```

{ Title: 'State wise International Airport count in South America', State:
"Peru", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Venezuela", Count: 3 },
{ Title: 'State wise International Airport count in South America', State:
"Bolivia", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Paraguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Uruguay", Count: 2 },
{ Title: 'State wise International Airport count in South America', State:
"Falkland Islands", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"French Guiana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Guyana", Count: 1 },
{ Title: 'State wise International Airport count in South America', State:
"Suriname", Count: 1 },
];
args.treemap.dataSource = data;
}
</script>

```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC TreeMap control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

### Data Binding

The TreeMap control supports data binding using the dataSource property.

### Populate data

The `dataSource` property accepts collection values as input. For example, a list of objects can be provided as input. Data can be given as either flat or hierarchical collection to the `dataSource` property.

<!-- markdownlint-disable MD036 -->

### Flat collection

The following code shows, how to bind a flat collection as data source to the TreeMap control.

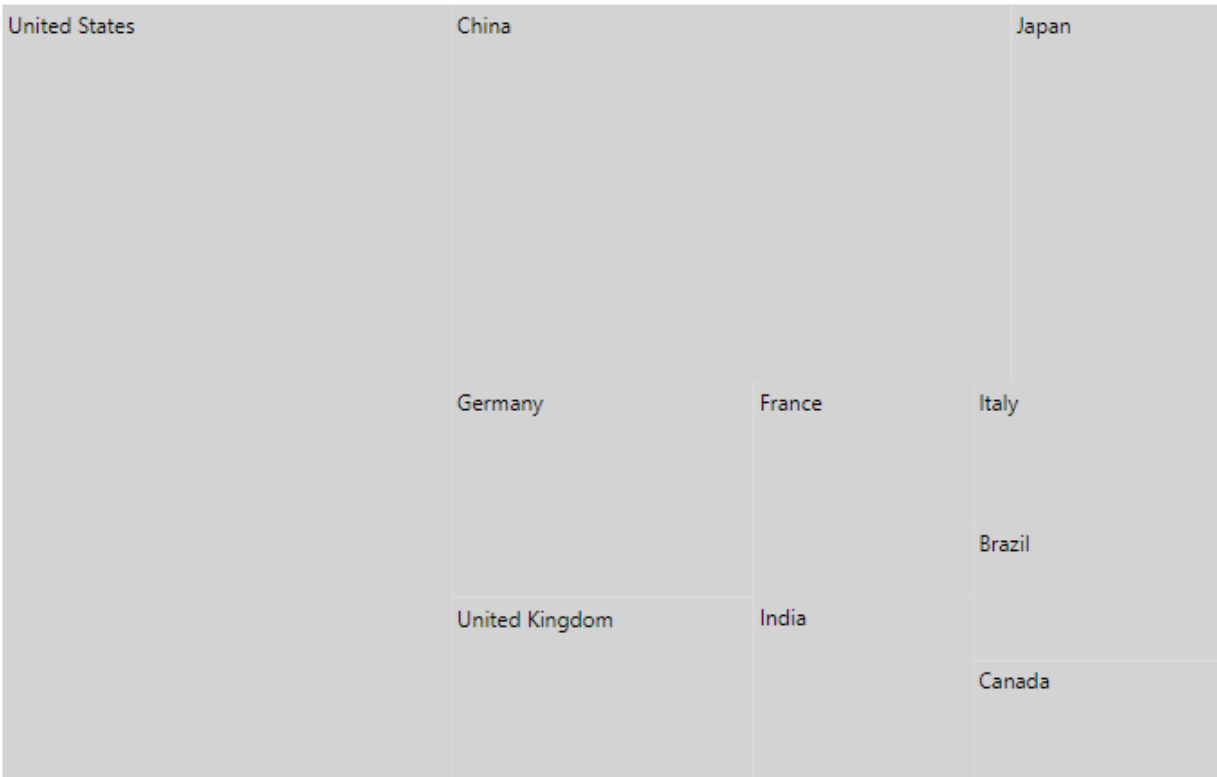
#### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("GDP").LeafItemSettings(leaf => leaf.LabelPath("State")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 }
</script>
```

#### FLAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```

}



### *Hierarchical collection*

The following code shows, how to bind a hierarchical collection as data source to the TreeMap control.

<!-- markdownlint-disable MD010 -->

### **CSHTML**

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("Population").
LeafItemSettings(leaf => leaf.LabelPath("Region").Fill("#0077b3").Border(new
TreeMapBorder { Color = "black", Width = 0.5 })).Levels(level =>
{
 level.GroupPath("Continent").Fill("#004466").Border(new
TreeMapBorder { Color = "black", Width = 0.5 }).Add();
 level.GroupPath("States").Fill("#0099e6").Border(new TreeMapBorder {
Color = "black", Width = 0.5 }).Add();
}).Render()
</div>
<script>
function load(args)
{
 var data = [{
 "Continent": [{
 "Name": "Africa",
 "Population": 1216130000,
 "States": [{
```

```

 "Name": "Eastern Africa",
 "Population": 410637987,
 "Region": [{
 "Name": "Ethiopia",
 "Population": 107534882
 }]
 },
 {
 "Name": "Middle Africa",
 "Population": 158562976,
 "Region": [{
 "Name": "Democratic, Republic of the Congo",
 "Population": 84004989
 }]
 }
]
}],
{
 "Continent": [{
 "Name": "Asia",
 "Population": 4436224000,
 "States": [{
 "Name": "Central Asia",
 "Population": 69787760,
 "Region": [{
 "Name": "Uzbekistan",
 "Population": 32364996
 }]
 }]
 },
 {
 "Name": "Eastern Asia",
 "Population": 1641908531,
 "Region": [{
 "Name": "China",
 "Population": 1415045928
 }]
 }
]
}],
{
 "Continent": [{
 "Name": "North America",
 "Population": 579024000,
 "States": [{
 "Name": "Central America",
 "Population": 174988756,
 "Region": [{
 "Name": "Mexico",
 "Population": 130759074
 }]
 }]
 },
 {
 "Name": "Northern America",
 "Population": 358593810,
 "Region": [{

```

```

 "Name": "U.S.",
 "Population": 3267667480
 }
]
 },
 {
 "Continent": [{
 "Name": "South America",
 "Population": 422535000,
 "States": [{
 "Name": "Brazil",
 "Population": 204519000
 }]
 }]
 },
 {
 "Continent": [{
 "Name": "Europe",
 "Population": 738849000,
 "States": [{
 "Name": "Eastern Europe",
 "Population": 291953328,
 "Region": [{
 "Name": "Russia",
 "Population": 143964709
 }]
 }]
 },
 {
 "Name": "Northern Europe",
 "Population": 103642971,
 "Region": [{
 "Name": "United Kingdom",
 "Population": 66573504
 }]
 }
]
 }
];
 args.treemap.dataSource = data;
}
</script>

```

## HIERACHICAL.CS

```

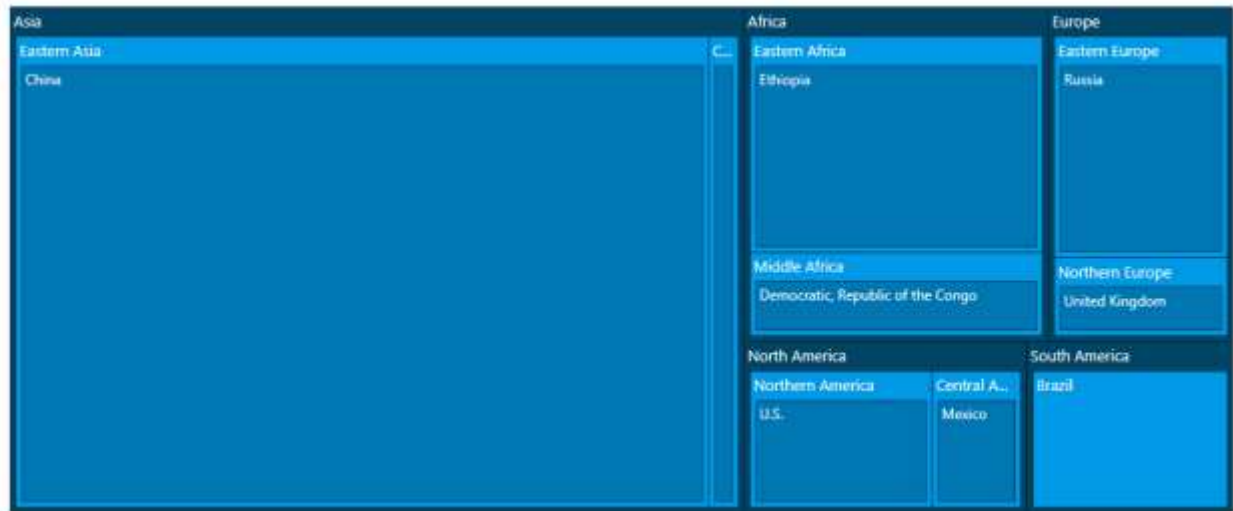
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers

```

```

{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



## Layout

Determine the visual representation of nodes belonging to all the TreeMap levels using the `layoutType` property.

### Types of layout

The available layout types are,

- Squarified
- SliceAndDiceVertical
- SliceAndDiceHorizontal
- SliceAndDiceAuto

### Squarified

The Squarified layout displays the nested rectangles based on aspect ratio in the TreeMap. The rectangles will be split based on the height and width of the parent. The default rendering type of layout is Squarified.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").WeightValuePath("GDP").Render(
)
</div>
<script>

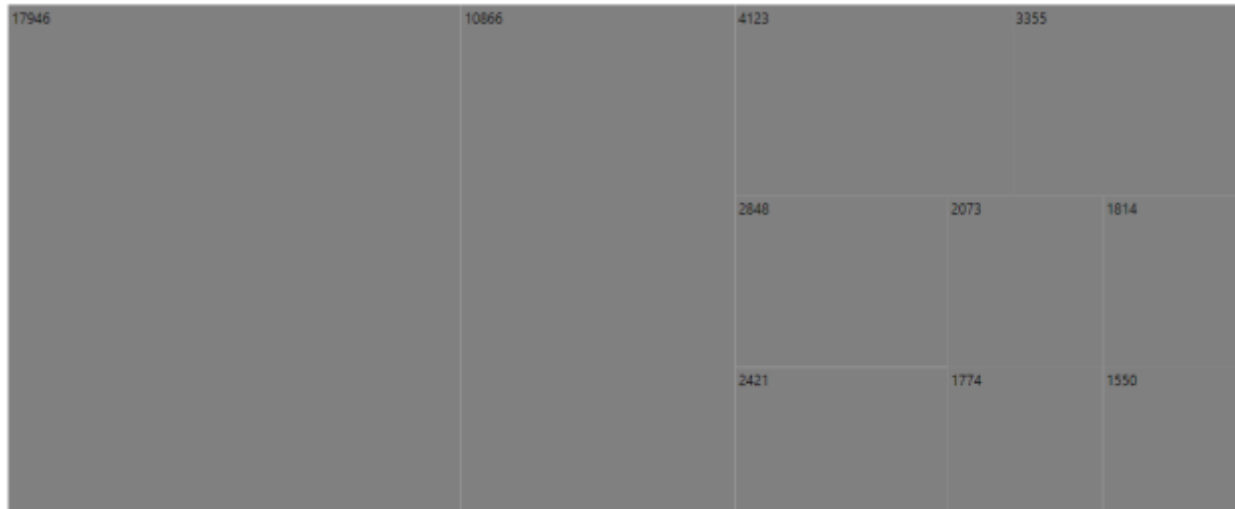
```

```
function load(args)
{
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
}
</script>
```

### SQUARIFIED.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```





### *SliceAndDiceVertical*

The **SliceAndDiceVertical** layout creates rectangles with high aspect ratio and displays items in a vertically sorted order.

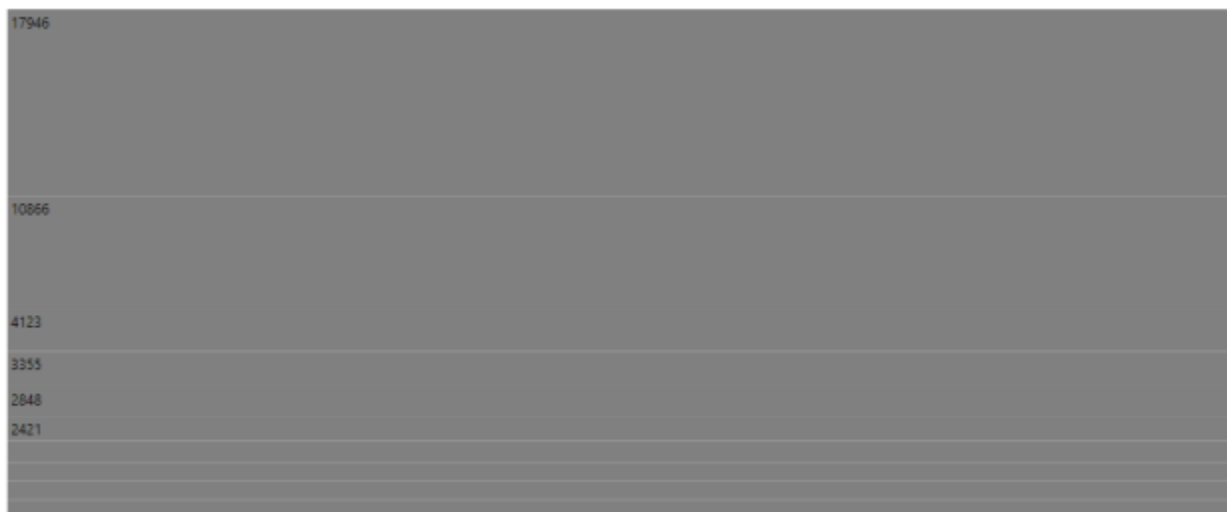
### **CSHTML**

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").LayoutType(Syncfusion.EJ2.Tree
Map.LayoutMode.SliceAndDiceVertical).WeightValuePath("GDP").Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
},
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
},
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 }
</script>
```

### **SLICE\_VERTICAL.CS**

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
```

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### *SliceAndDiceHorizontal*

The **SliceAndDiceHorizontal** layout creates rectangles with high aspect ratio and displays items in a horizontally sorted order.

### **CSHTML**

```
@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").LayoutType(Syncfusion.EJ2.Tree
 Map.LayoutMode.SliceAndDiceHorizontal).WeightValuePath("GDP").Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
```

```

 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
}
</script>

```

### SLICE\_HORIZONTAL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### *SliceAndDiceAuto*

The `SliceAndDiceAuto` layout creates rectangles with high aspect ratio and display items sorted both horizontally and vertically.

### CSHTML

```
@using Syncfusion.EJ2;
```

```

<div id="container">
@Html.EJS().TreeMap("container").Load("load").LayoutType(Syncfusion.EJ2.Tree
Map.LayoutMode.SliceAndDiceAuto).WeightValuePath("GDP").Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 }
</script>

```

### SLICE\_AUTO.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Right-to-left rendering

The TreeMap control supports right-to-left rendering for all its elements such as nodes, tooltip, data labels, and legends.

### Legend with Rtl support

If you set the `enableRtl` property to true, then the legend icon will be rendered on the right and the legend text will be rendered on the left of the legend icon.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").EnableRtl(true).WeightValuePath("count").LegendSettings(legend =>
 legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).LeafItemSettings(leaf => leaf.LabelPath("Car")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car:'Mustang', Brand:'Ford', count:232, color: '#71B081' },
 { Car:'EcoSport', Brand:'Ford', count:121, color:
 '#5A9A77' },
 { Car:'Swift', Brand:'Maruti', count:143, color:
 '#498770' },
 { Car:'Baleno', Brand:'Maruti', count:454, color:
 '#39776C' },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 ,
 color: '#266665' },
 { Car:'A3 Cabriolet', Brand:'Audi',count:123, color:
 '#124F5E' }
];
 args.treemap.dataSource = data;
 args.treemap.leafItemSettings.colorValuePath= 'color';
 }
</script>
```

**LEGEND-RTL.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

**Tooltip with Rtl support**

If the `enableRtl` property is set to true, the tooltip data will be rendered in reverse direction.

The following example shows the format of the tooltip.

**CSHTML**

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").WeightValuePath("count").tooltipSettings(to
oltip => tooltip.Visible(true).format('${count} :
${fruit}')).LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)

```

```

{
 var data = [
 { fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
 args.treemap.dataSource = data;
 args.treemap.palette=['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'];
}
</script>

```

### TOOLTIP-RTL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Treemap Item Rendering Direction

The direction of TreeMap item is **TopLeftBottomRight** by default and customize the rendering direction of the TreeMap item by setting the **RenderDirection** property.

The TreeMap can be rendered in the following directions:

- TopLeftBottomRight
- TopRightBottomLeft
- BottomRightTopLeft
- BottomLeftTopRight

The following example demonstrate how to render the treemap in the RTL direction with **TopLeftBottomRight**.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
 args.treemap.dataSource = data;
 args.treemap.palette=['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'];
 args.treemap.renderDirection = 'TopLeftBottomRight';
 }
</script>
```

### RENDER-DIRECTION-ONE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 }
```



```

public IActionResult Index()
{
 return View();
}

```



The following example demonstrate how to render the treemap in the RTL direction with `TopRightBottomLeft`.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 args.treemap.palette = ['#71B081', '#5A9A77', '#498770', '#39776C', '#266665', '#124F5E'];
 args.treemap.renderDirection = 'BottomRightTopLeft';
 }
</script>

```

**RENDER-DIRECTION-TWO.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



The following example demonstrate how to render the treemap in the RTL direction with BottomRightTopLeft.

**CSHTML**

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },

```

```

 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
 args.treemap.dataSource = data;
 args.treemap.palette=['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'];
 args.treemap.renderDirection = 'BottomRightTopLeft';
 }
</script>

```

### RENDER-DIRECTION-THREE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



The following example demonstrate how to render the treemap in the RTL direction with BottomLeftTopRight.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit:'Apple', count:5000 },
 { fruit:'Mango', count:3000 },
 { fruit:'Orange', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 },
 { fruit:'Melon', count:4500 }
];
 args.treemap.dataSource = data;
 args.treemap.palette=['#71B081', '#5A9A77', '#498770', '#39776C',
 '#266665', '#124F5E'];
 args.treemap.renderDirection = 'BottomLeftTopRight';
 }
</script>

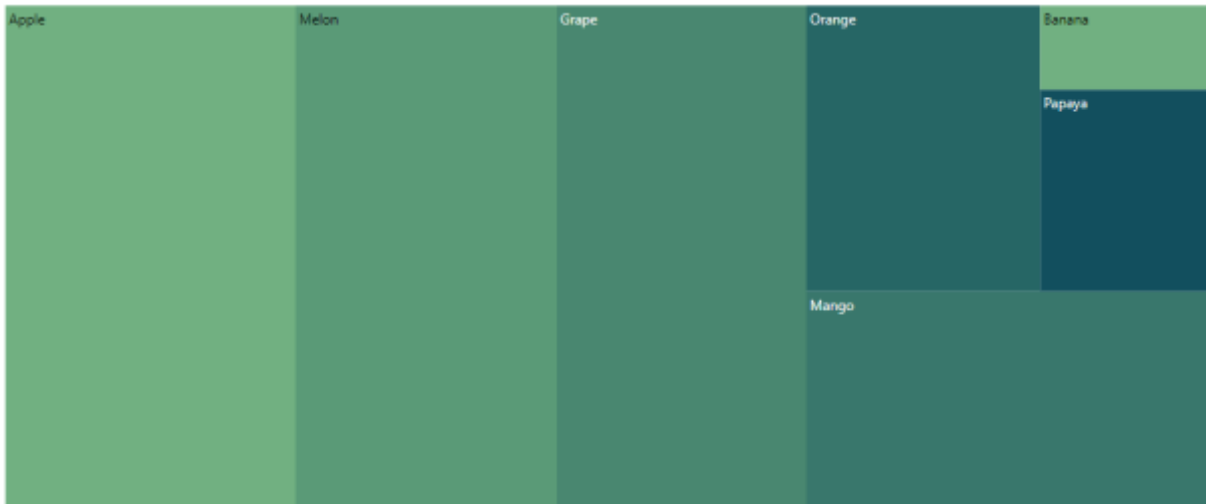
```

#### RENDER-DIRECTION-FOUR.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Leaf Item

A leaf item defines a visualized data element and does not contain child nodes but contains parent node if the levels are specified in the TreeMap.

### Leaf label

Label is represented by item name or value. Label will be appeared by specifying the `labelPath` property and customize the label style using the `labelStyle` property.

### CSHTML

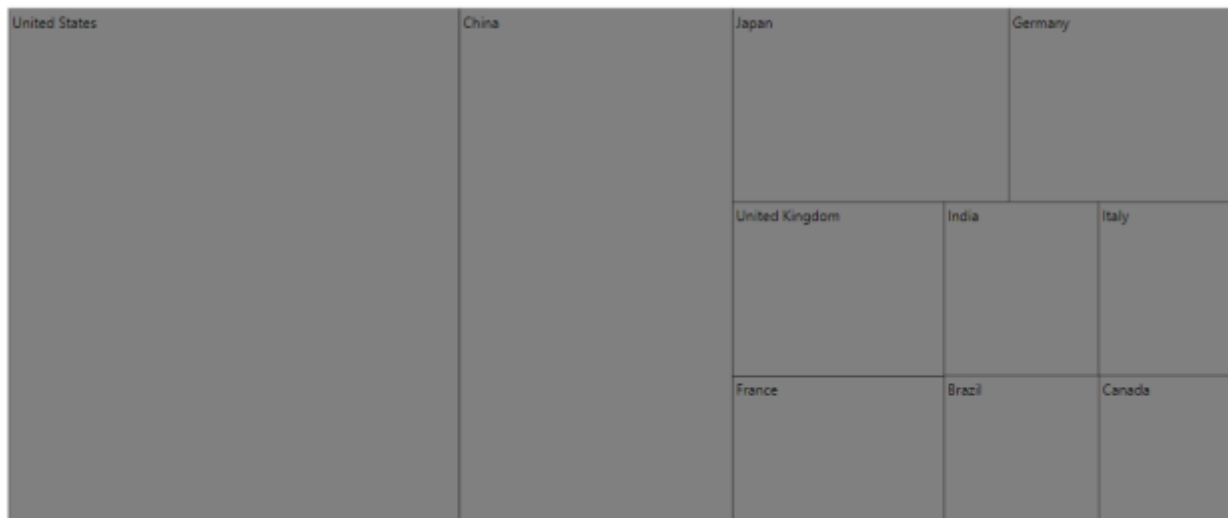
```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("GDP").LeafItem
mSettings(leaf => leaf.LabelPath("State").LabelStyle(new TreeMapFont { Color
= "#000000" })).Border(new TreeMapBorder { Color = "#000000", Width = 0.5
})).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
},
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
},
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 }
</script>
```

**LEAFLABEL.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



<!-- markdownlint-disable MD036 -->

*Label position and format*

Positioning the leaf item label using the `labelPosition` property and the text format can be customized by specifying data source properties name in the `labelFormat` property.

**CSHTML**

```

@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("GDP").LeafItemSettings(leaf =>
leaf.LabelPath("State").LabelPosition(Syncfusion.EJ2.TreeMap.LabelPosition.TopCenter).LabelFormat("${State}
${GDP} Trillion
({percentage}%)").Render()
</div>
<script>

```

```
function load(args)
{
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
}
</script>
```

## FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



<!-- markdownlint-disable MD036 -->

### Label template and position

Specifies the template of leaf item label and position of the template to be customized using `labelTemplate` and `templatePosition` properties.

### CSHTML

```
@using Syncfusion.EJ2;
<script id="labeltemp" type="text/x-template">
 <div>

 </div>
</script>
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").WeightValuePath("Gold").LeafItemSettings(leaf =>
 leaf.LabelPath("Sport").Fill("#993399").TemplatePosition(Syncfusion.EJ2.TreeMap.LabelPosition.Center).LabelTemplate("#labeltemp").Render()
 </div>
<script>
 function load(args)
 {
 var data = [
 { Sport: "Swimming", Gold: 16, GameImage: 'Swimming.svg',
 ItemHeight: "180px", ItemWidth: '180px' },
 { Sport: "Athletics", Gold: 13, GameImage: 'Athletics.svg',
 ItemHeight: "70px", ItemWidth: '70px' },
 { Sport: "Gymnastics", Gold: 4, GameImage: 'Gymnastics.svg',
 ItemHeight: "80px", ItemWidth: '80px' },
 { Sport: "Cycling", Gold: 2, GameImage: 'Cycling.svg',
 ItemHeight: "50px", ItemWidth: '50px' },
 { Sport: "Wrestling", Gold: 2, GameImage: 'Wrestling.svg',
 ItemHeight: "60px", ItemWidth: '50px' },
 { Sport: "Basketball", Gold: 2, GameImage: 'Basketball.svg',
 ItemHeight: "50px", ItemWidth: '50px' },
 { Sport: "Boxing", Gold: 1, GameImage: 'Boxing.svg', ItemHeight:
 "40px", ItemWidth: '30px' },
```



```

 { Sport: "Tennis", Gold: 1, GameImage: 'Tennis.svg', ItemHeight:
"40px", ItemWidth: '40px' },
 { Sport: "Judo", Gold: 1, GameImage: 'Judo.svg', ItemHeight:
"40px", ItemWidth: '40px' },
 { Sport: "Rowing", Gold: 1, GameImage: 'Rowing.svg', ItemHeight:
"40px", ItemWidth: '40px' },
 { Sport: "Shooting", Gold: 1, GameImage: 'Shooting.svg',
ItemHeight: "40px", ItemWidth: '40px' },
 { Sport: "Triathlon", Gold: 1, GameImage: 'Triathlon.svg',
ItemHeight: "40px", ItemWidth: '40px' },
 { Sport: "Water polo", Gold: 1, GameImage: 'Water polo.svg',
ItemHeight: "40px", ItemWidth: '40px' }
];
 args.treemap.dataSource = data;
}
</script>

```

### TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



<!-- markdownlint-disable MD036 -->

### Item gap

The `gap` property is used to separate an item from another item. Each item rectangle is split into equal space with specified gap.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("GDP").LeafItemSettings(leaf => leaf.LabelPath("State").Gap(20)).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 }
</script>
```

### GAP.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Levels

TreeMap supports **n** number of levels and each level is separated by using the `groupPath` property.

<!-- markdownlint-disable MD036 -->

### Group path

The `groupPath` property is used to separate each level of the TreeMap by specifying the property from the data source.

In the following example, three levels are added and each level is configured using the `groupPath` property.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").Levels(level =>
{
 level.GroupPath("Country").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobDescription").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobGroup").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
}).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
```

```

 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
];
 args.treemap.dataSource = data;
}
</script>

```

## GROUP PATH.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



<!-- markdownlint-disable MD036 -->

### Group gap

The **groupGap** property is used to separate an item from each group or another item to differentiate the levels mentioned in the TreeMap.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").Levels(level =>
{
 level.GroupPath("Country").GroupGap(10).Border(new TreeMapBorder {
Color = "black", Width = 0.5 }).Add();
 level.GroupPath("JobDescription").GroupGap(10).Border(new
TreeMapBorder { Color = "black", Width = 0.5 }).Add();
 level.GroupPath("JobGroup").GroupGap(10).Border(new TreeMapBorder {
Color = "black", Width = 0.5 }).Add();
}).Render()
</div>
<script>
function load(args)
{
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
```

```

 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
];
 args.treemap.dataSource = data;
}
</script>

```

### GROUP\_GAP.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



<!-- markdownlint-disable MD036 -->

### Header format and Alignment

Customize header using the `headerFormat` property in which fields are mapping from the `dataSource` and align header using the `headerAlignment` property.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").Levels(level =>
{
 level.GroupPath("Country").HeaderFormat("${Country}-
${EmployeesCount}").HeaderAlignment(Syncfusion.EJ2.TreeMap.Alignment.Center)
.Border(new TreeMapBorder { Color = "black", Width = 0.5 }).Add();
 level.GroupPath("JobDescription").HeaderFormat("${JobDescription}-
${EmployeesCount}").HeaderAlignment(Syncfusion.EJ2.TreeMap.Alignment.Far).Bo
rder(new TreeMapBorder { Color = "black", Width = 0.5 }).Add();
 level.GroupPath("JobGroup").HeaderFormat("${JobGroup}-
${EmployeesCount}").HeaderAlignment(Syncfusion.EJ2.TreeMap.Alignment.Near).B
order(new TreeMapBorder { Color = "black", Width = 0.5 }).Add();
}).Render()
</div>
<script>
function load(args)
{
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
```

```

 { Category: 'Employees', Country: 'India', JobDescription: 'HR
 Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
 'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
 'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
 Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
 'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
 'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
 'Marketing', EmployeesCount: 100 }
];
 args.treemap.dataSource = data;
}
</script>

```

## HEADER\_FORMAT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Header height and style

Customize the font color, family, weight, opacity and size using the `headerStyle`. Based on the font settings, the header height is given using the `headerHeight` property in `levels`.



**CSHTML**

```

@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").Levels(level =>
{
 level.GroupPath("Country").HeaderHeight(35).HeaderStyle(new
TreeMapFont { Size = "15px" }).Border(new TreeMapBorder { Color = "black",
Width = 0.5 }).Add();
 level.GroupPath("JobDescription").HeaderHeight(45).HeaderStyle(new
TreeMapFont { Size = "15px" }).Border(new TreeMapBorder { Color = "black",
Width = 0.5 }).Add();
 level.GroupPath("JobGroup").HeaderHeight(40).HeaderStyle(new
TreeMapFont { Size = "15px" }).Border(new TreeMapBorder { Color = "black",
Width = 0.5 }).Add();
}).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },

```

```

 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
];
 args.treemap.dataSource = data;
}
</script>

```

### HEADER HEIGHT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

### Header template and position

The TreeMap header supports to customize header of each item using the `headerTemplate` property. It uses Essential JS2 Template engine to render the elements. You can position the template using the `templatePosition` property.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").Palette(new
string[] { "#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").Levels(level =>
 {

 level.GroupPath("Country").HeaderTemplate("<div>{:Country}</div>").Templat
ePosition(Syncfusion.EJ2.TreeMap.LabelPosition.Center).Border(new
TreeMapBorder { Color = "black", Width = 0.5 }).Add();

 level.GroupPath("JobDescription").HeaderTemplate("<div>{:JobDescription}</
div>").TemplatePosition(Syncfusion.EJ2.TreeMap.LabelPosition.Center).Border(
new TreeMapBorder { Color = "black", Width = 0.5 }).Add();

 level.GroupPath("JobGroup").HeaderTemplate("<div>{:JobGroup}</div>").Templ
atePosition(Syncfusion.EJ2.TreeMap.LabelPosition.Center).Border(new
TreeMapBorder { Color = "black", Width = 0.5 }).Add();
 }).Render()
</div>

```

```

<script>
 function load(args)
 {
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
];
 args.treemap.dataSource = data;
 }
</script>

```

## TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers

```

```
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



## Color Mapping

Color mapping is used to customize the color for each group or item based on the specified types. The following options are available to customize the group and leaf items in the TreeMap.

### Range color mapping

Range color mapping is used to apply color to the items by giving specific ranges in the DataSource, and it should be specifying the data source properties to the `rangeColorValuePath`. The color mapping ranges to be specified in the `from` and `to` properties of the `colorMapping`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
 ColorValuePath("count").LeafItemSettings(leaf =>
 leaf.LabelPath("fruit")).Render()
</div>
<script>
 var treemap;
 function load(args) {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
]
 }
 treemap = new TreeMap({
 load: function (args) {
 load.apply(this, args);
 }
 });
 treemap.render($("#container"), true);
```

```
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 3000,
 color: 'orange'
 },
 {
 from: 3000,
 to: 5000,
 color: 'green'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
}
</script>
```

### RANGE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Equal color mapping

Equal color mapping is used to fill colors to each item by specifying equal value present in the data source, that can be specified in the `equalColorValuePath` property.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Equal
ColorValuePath("Brand").LeafItemSettings(leaf =>
leaf.LabelPath("Car")).Render()
</div>
<script>
 var treemap;
 function load(args) {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
]
 }
</script>
```

```

 };
 args.treemap.leafItemSettings.colorMapping = color;
}
</script>

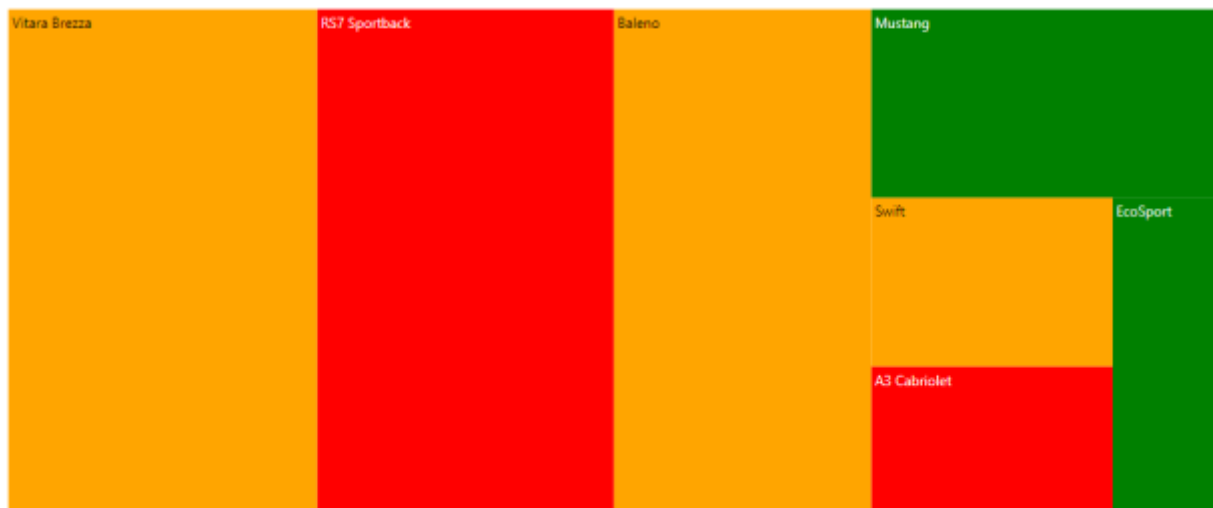
```

## EQUAL.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



## Desaturation color mapping

Desaturation color mapping is used to apply colors to the items based on `minOpacity` and `maxOpacity` properties in the `colorMapping`.

## CSHTML

```

@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
 ColorValuePath("count").LeafItemSettings(leaf =>
 leaf.LabelPath("fruit")).Render()

```

```
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 3000,
 minOpacity: 0.2,
 maxOpacity: 0.5,
 color: 'orange'
 },
 {
 from: 3000,
 to: 5000,
 minOpacity: 0.5,
 maxOpacity: 0.8,
 color: 'green'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
 }
</script>
```

### DESATURATION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```





### Palette color mapping

The palette color mapping is used to fill the color to each group or leaf item by given colors in the `palette` property.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] { "red",
"green"}).WeightValuePath("count").LeafItemSettings(leaf =>
leaf.LabelPath("Car")).Render()
</div>
<script>
 var treemap;
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 }
</script>
```

### PALLETE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
```

```
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Desaturation with multiple colors

Multiple colors are used as gradient effect to specific shapes based on the ranges in datasource. By using [color] property, you can set n number of colors.

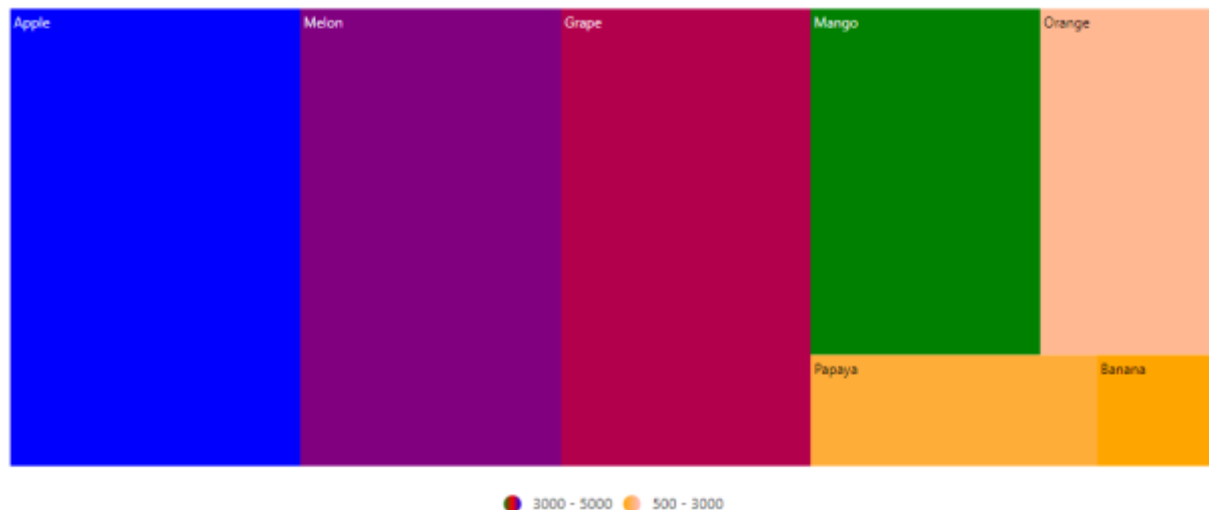
### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").Palette(new string[] { "red",
 "green" }).WeightValuePath("count").LeafItemSettings(leaf =>
 leaf.LabelPath("Car")).Render()
</div>
<script>
 var treemap;
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 }
</script>
```

```
}
</script>
```

### PALLETE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Color for items excluded from color mapping

Get the excluded ranges from data source using the color mapping and apply the specific color to those items, without specifying the `from` and `to` properties.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
 ColorValuePath("count").LeafItemSettings(leaf =>
 leaf.LabelPath("fruit")).Render()
</div>
<script>
```

```

var treemap;
function load(args) {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 2500,
 color: 'orange'
 },
 {
 from: 3000,
 to: 4000,
 color: 'green'
 },
 {
 color: 'red'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
}
</script>

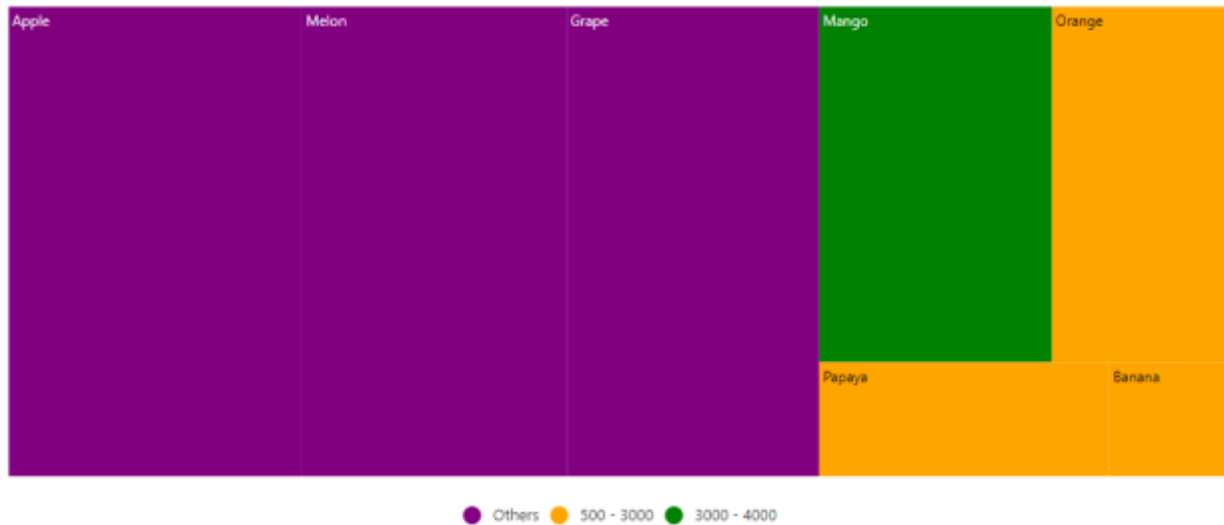
```

**EXCLUDECOLOR.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



Bind the colors to the items from data source

To set the color for each item from the data source, bind the data source property to the `colorValuePath`.

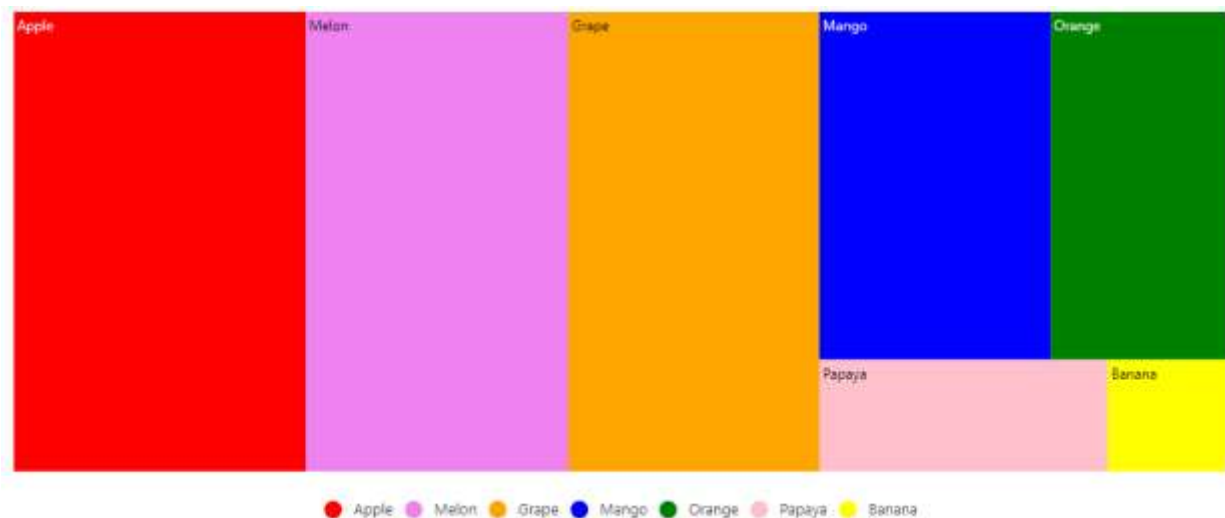
### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LeafItemSettings(leaf =>
leaf.LabelPath("fruit")).Render()
</div>
<script>
var treemap;
function load(args) {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 2500,
 color: ['orange', 'pink']
 },
 {
 from: 3000,
 to: 5000,
 color: ['red', 'blue']
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
}
```

```
}
</script>
```

### MULTIPLECOLOR.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Data Label

Data Labels are used to identify the name of items or groups in the TreeMap component. Data Labels will be shown by specifying the data source properties in the `labelPath` of the `leafItemSettings`.

### Format

Customize the labels for each item using the `labelFormat` property in the `leafItemSettings`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
```

```
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("Car").LabelFormat("${Car}-${Brand}")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 }
</script>
```

## FORMAT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Template

The template supports customizing labels of each leaf node using the `labelTemplate` property. It uses Essential JS2 template engine to render elements and the position of templates can be customize using the `templatePosition` property.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("Car").LabelTemplate("<div>{{:Car}}-{{:Brand}}</div>").TemplatePosition(Syncfusion.EJ2.TreeMap.LabelPosition.Center)).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
 { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
 { car:'RS7 Sportback', Brand:'Audi', count:523 }
];
 args.treemap.dataSource = data;
 }
</script>
```

### TEMPLATE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```



```

using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### IntersectAction

When the label size in each item exceeds the actual size, use the `interSeactAction` property in the `leafItemSettings` to customise the labels.

### CSHTML

```

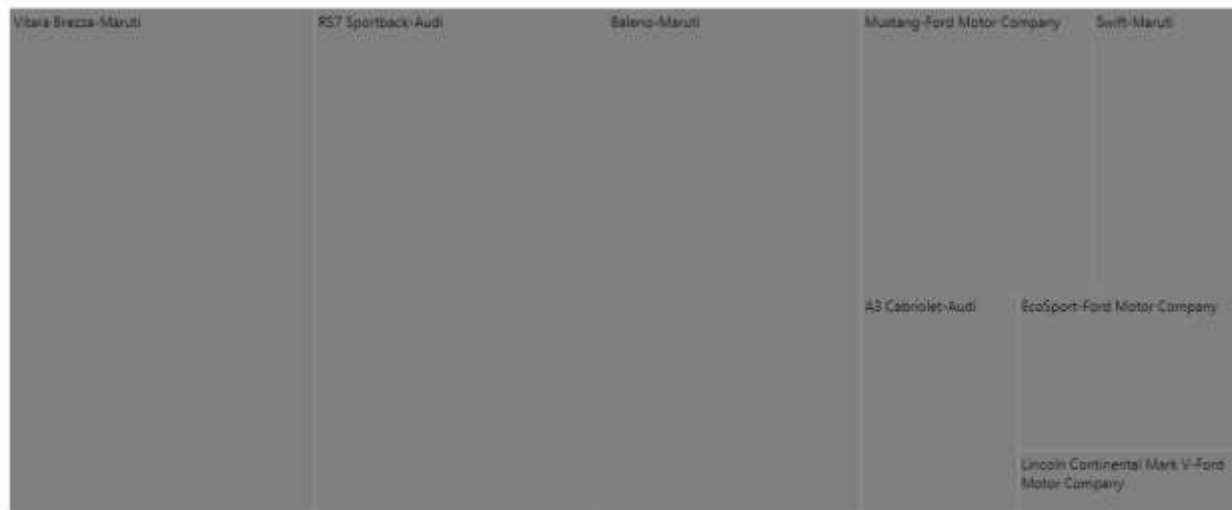
@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").LeafItemSettings(leaf => leaf.LabelPath("Car").LabelFormat("${Car}-${Brand}").InterSeactAction(Syncfusion.EJ2.TreeMap.LabelAlignment.WrapByWord)).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 }

```

```
args.treemap.dataSource = data;
}
</script>
```

## INTERSECTACTION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



## Legend

Legend is used to provide valuable information for interpreting what the TreeMap displays. The legends can be represented in various colors, shapes or other identifiers based on the data.

## Position and alignment

Legend position is used to place legend in various positions. Based on the legend position, the legend item will be aligned. For example, if the position is top or bottom, the legend items are placed by rows. If the position is left or right, the legend items are placed by columns.

The following options are available to customize the legend position:

- Top
- Bottom
- Left
- Right
- Float

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>
legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).Le
afItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 3000,
 color: 'orange'
 },
 {
 from: 3000,
 to: 5000,
 color: 'green'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
 }
</script>
```

### POSITION.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
```

```
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



Legend Alignment is used to align the legend items in specific location. The following options are available to customize the legend alignment:

- Near
- Center
- Far

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>
legend.Visible(true).Alignment(Syncfusion.EJ2.TreeMap.Alignment.Far)).LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
]
 }
}
```

```
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 3000,
 color: 'orange'
 },
 {
 from: 3000,
 to: 5000,
 color: 'green'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
}
</script>
```

### ALIGN.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Legend mode

The TreeMap control supports two different types of legend rendering modes such as **Default** and **Interactive**.

<!-- markdownlint-disable MD036 -->

### Default mode

In default mode, the legends have symbols with legend labels that are used to identify the items in the TreeMap.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Equal
ColorValuePath("Brand").LegendSettings(legend =>
legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).Le
afItemSettings(leaf => leaf.LabelPath("Car")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 value: 'Ford',
 color: 'green'
 },
],
 {
```

```

 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
];
args.treemap.leafItemSettings.colorMapping = color;
var legend = { color: "black", width: 2 };
args.treemap.legendSettings.border = legend;
}
</script>

```

### DEFAULT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



<!-- markdownlint-disable MD036 -->

*Interactive mode*

The legends can be made interactive with an arrow mark that indicates exact range color in the legend when the mouse hovers on the TreeMap item. Enable this option by setting the `mode` property in the `legendSettings` to **Interactive**.

**CSHTML**

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Equal
ColorValuePath("Brand").LegendSettings(legend =>
legend.Visible(true).Mode(Syncfusion.EJ2.TreeMap.LegendMode.Interactive).Pos
ition(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).LeafItemSettings(leaf =>
leaf.LabelPath("Car")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
 var legend = { color: "black", width: 2 };
 args.treemap.legendSettings.border = legend;
 }
</script>
```

**INTERACTIVE.CS**

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
```



```
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Legend size

Customize the legend size by modifying the `height` and `width` properties in the `legendSettings`. It accepts values in both percentage and pixel.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Equal
ColorValuePath("Brand").LegendSettings(legend =>
legend.Visible(true).Height("50px").Width("200px").Position(Syncfusion.EJ2.T
reeMap.LegendPosition.Top)).LeafItemSettings(leaf =>
leaf.LabelPath("Car")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car:'Mustang', Brand:'Ford', count:232 },
 { Car:'EcoSport', Brand:'Ford', count:121 },
 { Car:'Swift', Brand:'Maruti', count:143 },
 { Car:'Baleno', Brand:'Maruti', count:454 },
 { Car:'Vitara Brezza', Brand:'Maruti', count:545 },
]
 }
</script>
```

```

 { Car:'A3 Cabriolet', Brand:'Audi',count:123 },
 { car:'RS7 Sportback', Brand:'Audi', count:523 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
 color: 'orange'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
 var legend = { color: "black", width: 2 };
 args.treemap.legendSettings.border = legend;
 }
</script>

```

**SIZE.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Paging support

TreeMap support legend paging, if the legend items cannot be placed within the provided **height** and **width** of the legend.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Equal
ColorValuePath("Brand").LegendSettings(legend =>
legend.Visible(true).Height("50px").Width("100px")).LeafItemSettings(leaf =>
leaf.LabelPath("Car")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { Car: 'Mustang', Brand: 'Ford', count: 232 },
 { Car: 'EcoSport', Brand: 'Ford', count: 121 },
 { Car: 'Swift', Brand: 'Maruti', count: 143 },
 { Car: 'Baleno', Brand: 'Maruti', count: 454 },
 { Car: 'Vitara Brezza', Brand: 'Maruti', count: 545 },
 { Car: 'A3 Cabriolet', Brand: 'Audi', count: 123 },
 { Car: 'RS7 Sportback', Brand: 'Audi', count: 523 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 value: 'Ford',
 color: 'green'
 },
 {
 value: 'Audi',
 color: 'red'
 },
 {
 value: 'Maruti',
```

```

 color: 'orange'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
 var legend = { color: "black", width: 2 };
 args.treemap.legendSettings.border = legend;
}
</script>

```

## PAGING.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



## Legend for items excluded from color mapping

Based on the mapping ranges in the data source, get the excluded ranges from the color mapping, and show the legend with the excluded range values that are bound to the specific legend.

## CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>

```

```

legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 2500,
 color: 'orange'
 },
 {
 from: 3000,
 to: 4000,
 color: 'green'
 },
 {
 color: 'violet'
 }
];
 args.treemap.leafItemSettings.colorMapping = color;
 }
</script>

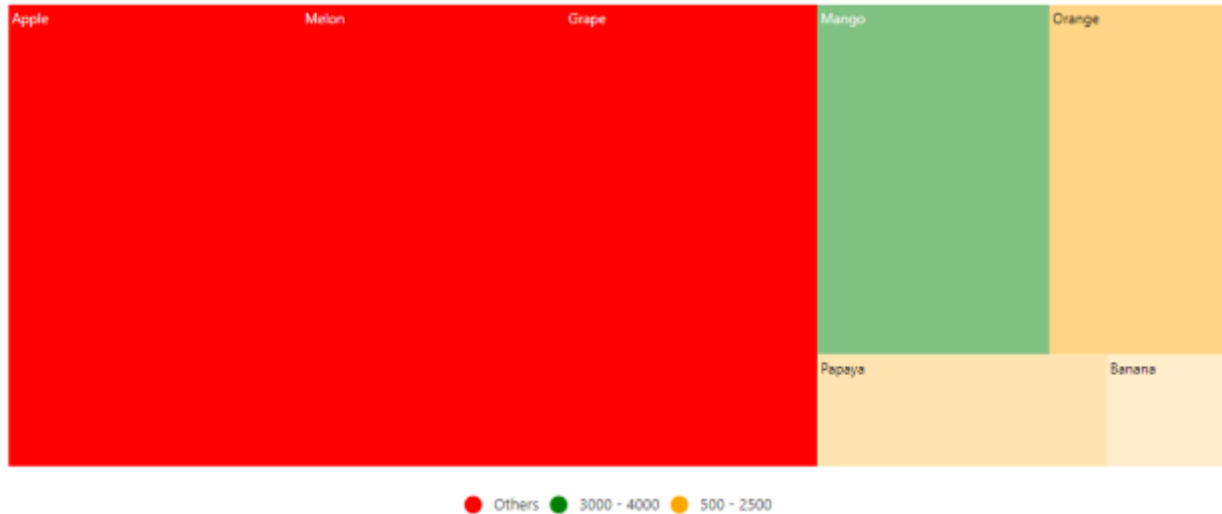
```

**EXCLUDELEGEND.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Hide desired legend items

To enable or disable the desired legend item for each color mapping, set the `showLegend` property to `true` in the `colorMapping`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>
legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).Le
afItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
function load(args)
{
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 var color = [
 {
 from: 500,
 to: 3000,
 color: 'orange',
 showLegend: true,
 },
 {
 from: 3000,
 to: 5000,
 color: 'green',

```

```

 showLegend: false,
 }
];
args.treemap.leafItemSettings.colorMapping = color;
}
</script>

```

### HIDELEGEND.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Hide legend items based data source value

To enable or disable the legend visibility for each item through the data source, bind the appropriate data source field name to `showLegendPath` property in the `legendSettings`.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>
 legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).Le
afItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>

```

```

function load(args)
{
 var data = [
 { fruit: 'Apple', count: 5000 , color: 'red', visibility: true },
 { fruit: 'Mango', count: 3000, color: 'blue' , visibility:
true},
 { fruit: 'Orange', count: 2300,color: 'green' , visibility:
false},
 { fruit: 'Mango', count: 500, color: 'pink' , visibility: true
},
 { fruit: 'Apple', count: 4300, color: 'yellow' , visibility:
false },
 { fruit: 'Papaya', count: 1200 color: 'orange', visibility:
true},
 { fruit: 'Melon', count: 4500, color: 'violet', visibility:
false }
];
 args.treemap.dataSource = data;
 args.treemap.colorValuePath = color;
 args.treemap.LegendSettings.visible = true;
 args.treemap.LegendSettings.showLegendPath = visibility;
 args.treemap.palette = [];
}
</script>

```

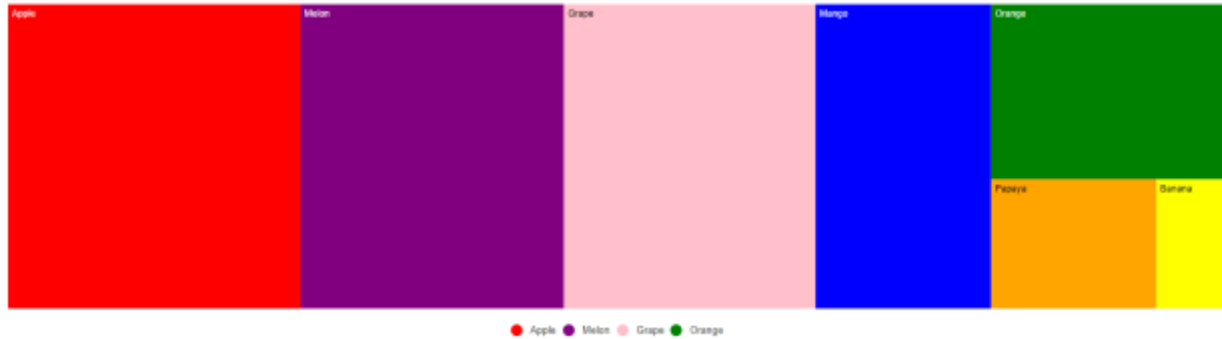
**HIDELEGENDBASEDDS.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```





Bind legend item text from data source

To show the legend item text from the data source, bind the property name from data source to the `valuePath` property in the `legendSettings`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>
legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).Le
afItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 , color: 'red' },
 { fruit: 'Mango', count: 3000, color: 'blue' },
 { fruit: 'Orange', count: 2300,color: 'green'},
 { fruit: 'Banana', count: 500, color: 'pink' },
 { fruit: 'Grape', count: 4300, color: 'yellow' },
 { fruit: 'Papaya', count: 1200 color: 'orange'},
 { fruit: 'Melon', count: 4500, color: 'violet' }
];
 args.treemap.dataSource = data;
 args.treemap.colorValuePath = color;
 args.treemap.LegendSettings.visible = true;
 args.treemap.LegendSettings.valuePath = fruit;
 args.treemap.palette = [];
 }
</script>
```

### BINDLEGENDTEXT.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
```

```
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Hide duplicate legend items

To enable or disable the duplicate legend items, set the `removeDuplicateLegend` property to **true** in the `legendSettings`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").Range
ColorValuePath("count").LegendSettings(legend =>
legend.Visible(true).Position(Syncfusion.EJ2.TreeMap.LegendPosition.Top)).Le
afItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 , color: 'red' },
 { fruit: 'Mango', count: 3000, color: 'blue' },
 { fruit: 'Orange', count: 2300,color: 'green'},
 { fruit: 'Mango', count: 500, color: 'pink' },
 { fruit: 'Apple', count: 4300, color: 'yellow' },
 { fruit: 'Papaya', count: 1200 color: 'orange'},
 { fruit: 'Melon', count: 4500, color: 'violet' }
];
 args.treemap.dataSource = data;
 args.treemap.colorValuePath = color;
 args.treemap.LegendSettings.visible = true;
 args.treemap.LegendSettings.valuePath = fruit;
 args.treemap.LegendSettings.removeDuplicateLegend = true;
 }
</script>
```

```

 args.treemap.palette = [];
 }
</script>

```

### DUPLICATELEGEND.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Legend Responsiveness

Use a responsive legend that switches positions between the right and the bottom based on the available height and width. To enable the responsive legend, set the **position** property to **Auto** in the **legendSettings** and the legend position is changed based on the available height and width.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load")..WeightValuePath("count").LegendSettings(legend =>
 legend.Visible(true).position('Top')).LeafItemSettings(leaf =>
 leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {

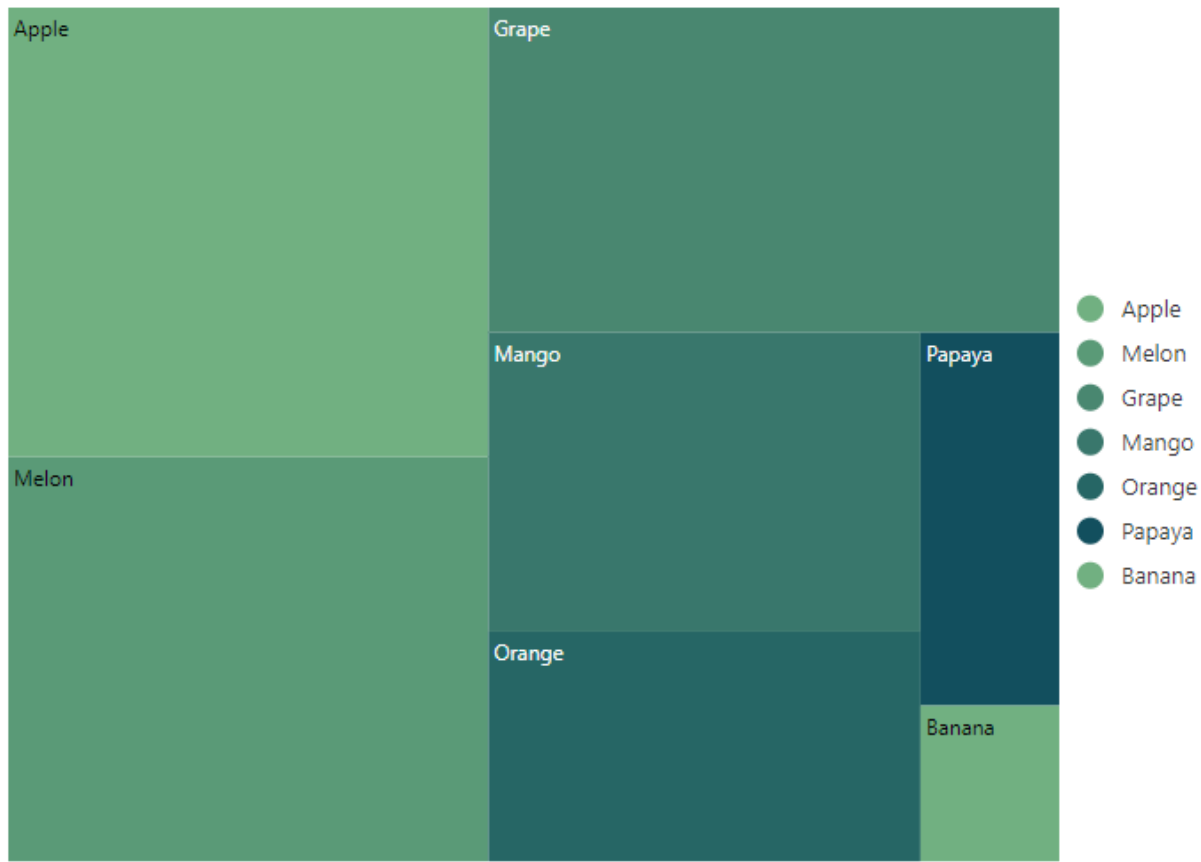
```

```
var data = [
 { fruit:'Apple', count:5000 },
 { fruit:'Grape', count:3000 },
 { fruit:'Apple', count:2300 },
 { fruit:'Banana', count:500 },
 { fruit:'Grape', count:4300 },
 { fruit:'Papaya', count:1200 }
];
args.treemap.dataSource = data;

args.treemap.leafItemSettings.colorMapping = color;
var size = { width: 700, height: 500 };
args.treemap.legendSettings.size = size;
}
</script>
```

### LEGEND-RESPONSIVE.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}
```



### Drill-down

The TreeMap supports drill-down to expose the hierarchy, achieved by clicking a node. If an item is clicked in the TreeMap, it will be moved to the next level or sub level hierarchy and returned back to the previous level by clicking the node.

#### Perform drill-down action

The TreeMap items can be drilled by setting the `enableDrillDown` property to **true**.

#### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").EnableDrillDown(true).Levels(level =>
{
 level.GroupPath("Country").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobDescription").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobGroup").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
}).Render()
</div>
<script>
 function load(args)
 {
```

```

var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
];
args.treemap.dataSource = data;
}
</script>

```

### DRILLDOWN.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {

```

```

public IActionResult Index()
{
 return View();
}

```



### On-demand data loading

All the child items are rendered during the normal drill-down process, and visible at the initial rendering of the TreeMap. But on-demand data loading, it will not render child items at initial rendering, and child nodes will be rendered during the drill-down process by setting the `drillDownView` property to `true`.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
"#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
}).WeightValuePath("EmployeesCount").EnableDrillDown(true).DrillDownView(true)
).Levels(level =>
{
 level.GroupPath("Country").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobDescription").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobGroup").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
}).Render()
</div>
<script>
function load(args)
{
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },

```

```

 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
 };
 args.treemap.dataSource = data;
}
</script>

```

## DRILLDOWN-DEMAND.CS

```

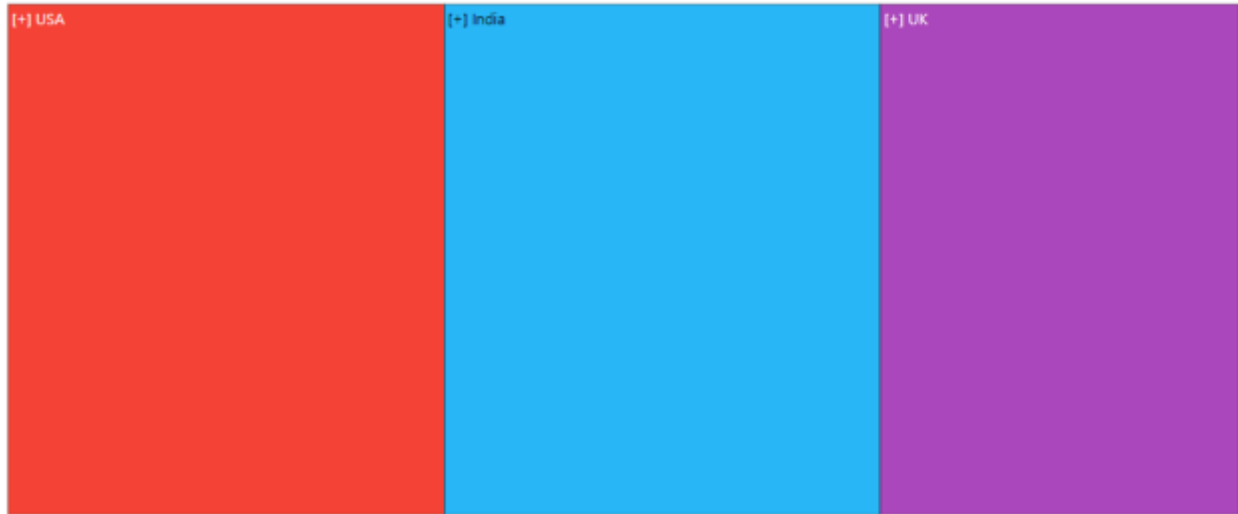
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



```
}

```



### Breadcrumb support

TreeMap items are drilled, up to any level of parent using breadcrumb navigation and the level from root parent to current level is displayed at the top of item layout. It can be enabled by using the `enableBreadcrumb` property to **true** and customize the breadcrumb connector using the `breadcrumbConnector` property. By default, `-(hyphen)` is the connector.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
 @Html.EJS().TreeMap("container").Load("load").Palette(new string[] {
 "#f44336", "#29b6f6", "#ab47bc", "#ffc107", "#5c6bc0", "#009688"
 }).WeightValuePath("EmployeesCount").EnableDrillDown(true).enableBreadcrumb(
 true).breadcrumbConnector(' -> ').Levels(level =>
 {
 level.GroupPath("Country").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobDescription").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 level.GroupPath("JobGroup").Border(new TreeMapBorder { Color =
"black", Width = 0.5 }).Add();
 }).Render()
 </div>
 <script>
 function load(args)
 {
 var data = [
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Marketing', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'USA', JobDescription:
'Management', EmployeesCount: 80 },
]
 }
 </script>

```

```

 { Category: 'Employees', Country: 'India', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 100 },
 { Category: 'Employees', Country: 'India', JobDescription: 'HR
Executives', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'India', JobDescription:
'Accounts', EmployeesCount: 40 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Executive', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Sales', JobGroup: 'Analyst', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Marketing', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 80 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Management', EmployeesCount: 10 },
 { Category: 'Employees', Country: 'Germany', JobDescription:
'Accounts', EmployeesCount: 20 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 30 },
 { Category: 'Employees', Country: 'UK', JobDescription: 'HR
Executives', EmployeesCount: 50 },
 { Category: 'Employees', Country: 'UK', JobDescription:
'Accounts', EmployeesCount: 60 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Technical', JobGroup: 'Testers', EmployeesCount: 70 },
 { Category: 'Employees', Country: 'France', JobDescription:
'Marketing', EmployeesCount: 100 }
];
 args.treemap.dataSource = data;
}
</script>

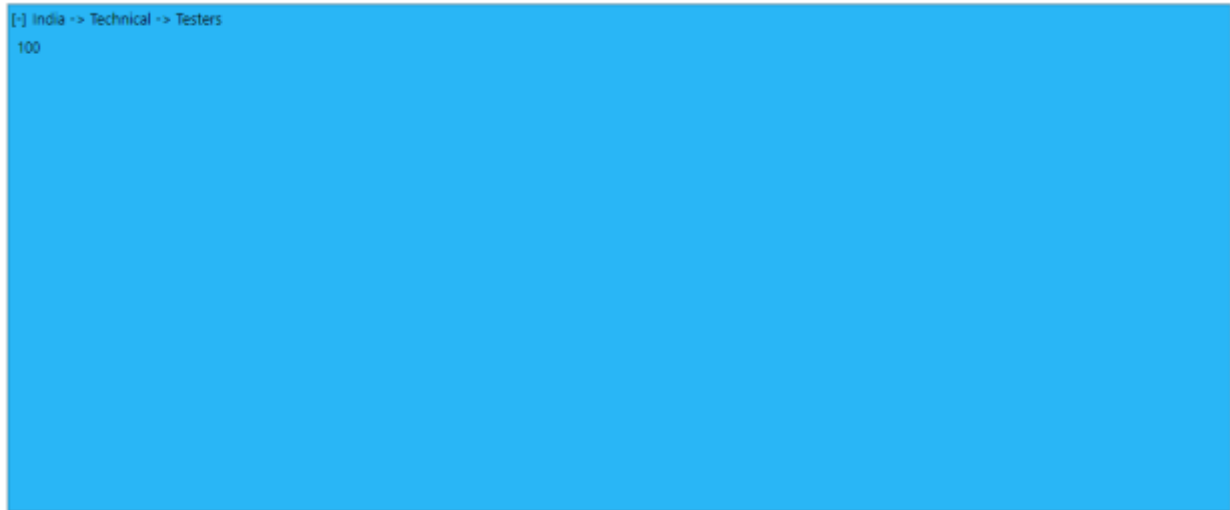
```

### DRILLDOWN-BREADCRUMB.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



## Tooltip

Tooltip is used to display details about the items in the TreeMap. When space constraints prevent us from displaying the information using Data Labels, the tooltip comes in handy.

### Default tooltip

The tooltip is not visible by default, to make it visible, set the `visible` property in the `tooltipSettings` to `true`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").TooltipSettings(tool => tool.Visible(true)).LeafItemSettings(leaf =>
leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
 }
</script>
```

### TOOLTIP.CS

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
```

```

using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Format tooltip

The tooltip content is displayed by default based on the `weightValuePath`. In addition, to show more information in the tooltip, use the `format` property and define field from the data source as `${datafield}`.

### CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").TooltipSettings(tool => tool.Visible(true).Format("Name:${fruit} - TotalCount:${count}")).LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 }

```

```

 args.treemap.dataSource = data;
 }
</script>

```

## FORMAT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



## Tooltip template

Tooltip can be rendered as a custom component using the `template` property in the `tooltipSettings` which accepts one or more UI elements as an input, that can be rendered as a part of the tooltip rendering. You can use `${datafield}` as placeholder in HTML element to display the values from data source.

## CSHTML

```

@using Syncfusion.EJ2;
<div id="container">

@Html.EJS().TreeMap("container").Load("load").WeightValuePath("count").TooltipSettings(tool => tool.Visible(true).Template("<div><p>Name: ${fruit}</p><p>Total Count: ${count}</p></div>")).LeafItemSettings(leaf => leaf.LabelPath("fruit")).Render()
</div>
<script>
 function load(args)

```

```

{
 var data = [
 { fruit: 'Apple', count: 5000 },
 { fruit: 'Mango', count: 3000 },
 { fruit: 'Orange', count: 2300 },
 { fruit: 'Banana', count: 500 },
 { fruit: 'Grape', count: 4300 },
 { fruit: 'Papaya', count: 1200 },
 { fruit: 'Melon', count: 4500 }
];
 args.treemap.dataSource = data;
}
</script>

```

### TEMPLATE.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Selection and Highlight

#### Selection

Selection is used to select a particular group or item to differentiate from other items. Each item or each group can be selected and deselected while interacting with the items. The corresponding Treemap items are also selected while tapping a specific legend item, and vice versa.

The `fill` property is used to change the selected item color. The `color` and the `width` properties are used to customize the selected item border, and the selection is enabled by using the `enable` property to `true` in the `selectionSettings`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("sales").Level
s(level =>
 {
 level.GroupPath("dataType").Fill("#c5e2f7").HeaderStyle(new
TreeMapFont { Size = "16px"
 }).HeaderAlignment(Syncfusion.EJ2.TreeMap.Alignment.Center).GroupGap(5).Add(
);

level.GroupPath("product").Fill("#a4d1f2").HeaderAlignment(Syncfusion.EJ2.Tr
eeMap.Alignment.Center).GroupGap(2).Add();
 }).LeafItemSettings(leaf =>
leaf.LabelPath("type").Fill("#8ebfe2").LabelPosition(Syncfusion.EJ2.TreeMap.
LabelPosition.Center)).SelectionSettings(sel =>
sel.Enable(true).Fill("#58a0d3").Opacity("1")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { dataType: "Import", type: "Animal products", product: "2010",
sales: 20839332874 },
 { dataType: "Import", type: "Animal products", product: "2011",
sales: 23098635589 },
 { dataType: "Import", type: "Chemical products", product:
"2010", sales: 141637951510 },
 { dataType: "Import", type: "Chemical products", product:
"2011", sales: 161550338209 },
 { dataType: "Import", type: "Base metals", product: "2010",
sales: 86079439944 },
 { dataType: "Import", type: "Base metals", product: "2011",
sales: 103821671535 },
 { dataType: "Import", type: "Textile articles", product: "2010",
sales: 97126140830 },
 { dataType: "Import", type: "Textile articles", product: "2011",
sales: 104980750811 },
 { dataType: "Export", type: "Animal products", product: "2010",
sales: 15845503378 },
 { dataType: "Export", type: "Animal products", product: "2011",
sales: 20650111620 },
 { dataType: "Export", type: "Chemical products", product:
"2010", sales: 136100054087 },
 { dataType: "Export", type: "Chemical products", product:
"2011", sales: 146341672411 },
 { dataType: "Export", type: "Base metals", product: "2010",
sales: 59060592813 },
 { dataType: "Export", type: "Base metals", product: "2011",
sales: 71785882641 },
]
 }
}
```

```

 { dataType: "Export", type: "Textile articles", product: "2010",
sales: 20982380561 },
 { dataType: "Export", type: "Textile articles", product: "2011",
sales: 26016143783 }
];
 args.treemap.dataSource = data;
}
</script>

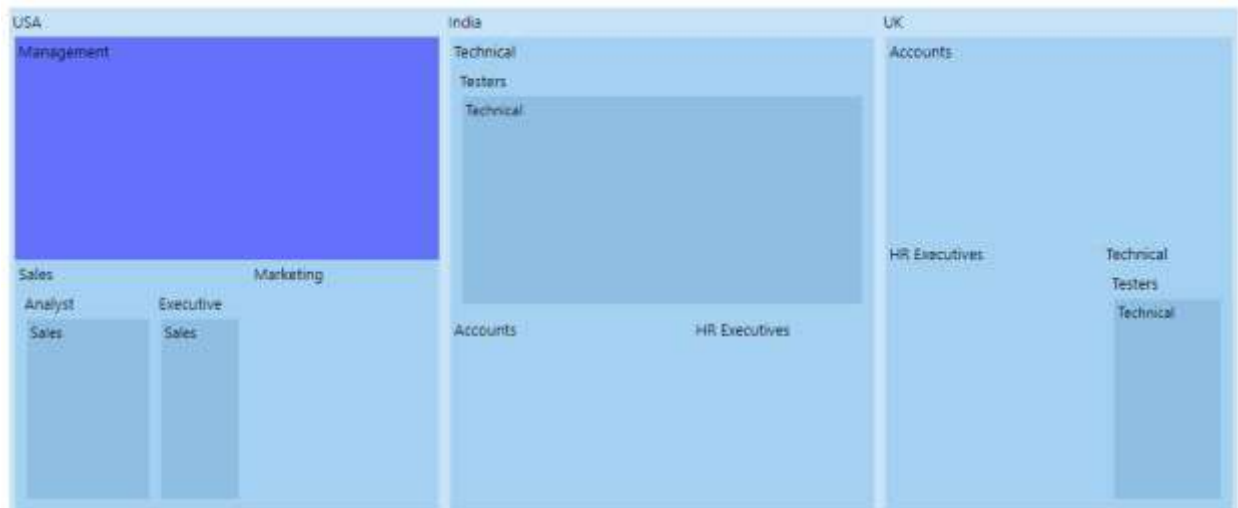
```

### SELECTION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Highlight

Highlight is used to highlight an item or group from other items or groups. Each item or each group can be highlighted by hovering the mouse over the items. The corresponding Treemap items are also be highlighted while hovering over a specific legend item, and vice versa.



The `fill` property is used to change the highlighted item color. The `color` and the `width` properties are used to customize the highlighted item border, and the highlight is enabled by setting the `enable` property to `true` in the `highlightSettings`.

### CSHTML

```
@using Syncfusion.EJ2;
<div id="container">
@Html.EJS().TreeMap("container").Load("load").WeightValuePath("sales").Level
s(level =>
 {
 level.GroupPath("dataType").Fill("#c5e2f7").HeaderStyle(new
TreeMapFont { Size = "16px"
 }).HeaderAlignment(Syncfusion.EJ2.TreeMap.Alignment.Center).GroupGap(5).Add(
);

 level.GroupPath("product").Fill("#a4d1f2").HeaderAlignment(Syncfusion.EJ2.Tr
eeMap.Alignment.Center).GroupGap(2).Add();
 }).LeafItemSettings(leaf =>
leaf.LabelPath("type").Fill("#8ebfe2").LabelPosition(Syncfusion.EJ2.TreeMap.
LabelPosition.Center)).HighlightSettings(high =>
high.Enable(true).Fill("#71b0dd").Opacity("1")).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { dataType: "Import", type: "Animal products", product: "2010",
sales: 20839332874 },
 { dataType: "Import", type: "Animal products", product: "2011",
sales: 23098635589 },
 { dataType: "Import", type: "Chemical products", product:
"2010", sales: 141637951510 },
 { dataType: "Import", type: "Chemical products", product:
"2011", sales: 161550338209 },
 { dataType: "Import", type: "Base metals", product: "2010",
sales: 86079439944 },
 { dataType: "Import", type: "Base metals", product: "2011",
sales: 103821671535 },
 { dataType: "Import", type: "Textile articles", product: "2010",
sales: 97126140830 },
 { dataType: "Import", type: "Textile articles", product: "2011",
sales: 104980750811 },
 { dataType: "Export", type: "Animal products", product: "2010",
sales: 15845503378 },
 { dataType: "Export", type: "Animal products", product: "2011",
sales: 20650111620 },
 { dataType: "Export", type: "Chemical products", product:
"2010", sales: 136100054087 },
 { dataType: "Export", type: "Chemical products", product:
"2011", sales: 146341672411 },
 { dataType: "Export", type: "Base metals", product: "2010",
sales: 59060592813 },
 { dataType: "Export", type: "Base metals", product: "2011",
sales: 71785882641 },
]
 }
}
```

```

 { dataType: "Export", type: "Textile articles", product: "2010",
sales: 20982380561 },
 { dataType: "Export", type: "Textile articles", product: "2011",
sales: 26016143783 }
];
 args.treemap.dataSource = data;
}
</script>

```

### HIGHLIGHT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```



### Print and Export

#### Print

To use the print functionality, we should set the [allowPrint](#) property to **true**. The rendered treemap can be printed directly from the browser by calling the method [print](#).

### CSHTML

```

@using Syncfusion.EJ2;">
<div>
 <button id="togglebtn">Print</button>
</div>
<div id="container">

@Html.EJS().TreeMap("container").Load("load").AllowPrint(true).WeightValuePa
th("GDP").LeafItemSettings(leaf =>
leaf.LabelPath("State").LabelFormat("${State}
${GDP}
Trillion
({percentage} %)").LabelStyle(new TreeMapFont { Color =
"#000000" }).Border(new TreeMapBorder { Color = "#000000", Width = 0.5
})).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
},
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
},
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 document.getElementById('togglebtn').onclick = () => {
 args.treemap.print();
 };
 }
</script>

```

**PRINT.CS**

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

```
}

```

## Export

### Image Export

To use the image export functionality, we should set the [allowImageExport](#) property to **true**. The rendered treemap can be exported as an image using the [export](#) method. The method requires two parameters: image type and file name. The treemap can be exported as an image in the following formats.

- JPEG
- PNG
- SVG

## CSHTML

```
@using Syncfusion.EJ2;">
<div>
 <button id="togglebtn">Export</button>
</div>
<div id="container">

@Html.EJS().TreeMap("container").Load("load").AllowImageExport(true).WeightV
aluePath("GDP").LeafItemSettings(leaf =>
leaf.LabelPath("State").LabelFormat("${State}
${GDP}
Trillion
({percentage} %)").LabelStyle(new TreeMapFont { Color =
"#000000" }).Border(new TreeMapBorder { Color = "#000000", Width = 0.5
})).Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
},
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
},
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 document.getElementById('togglebtn').onclick = () => {
 args.treemap.export('PNG', 'TreeMap');
 };
 }
</script>
```

## EXPORT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

We can get the image file as base64 string for the JPEG and PNG formats. The treemap can be exported to image as a base64 string using the [export](#) method. There are four parameters required: image type, file name, orientation of the exported PDF document which must be set as **null** for image export and finally **allowDownload** which should be set as **false** to return base64 string.

### CSHTML

```

@using Syncfusion.EJ2;">
<div>
 <button id="togglebtn">Export</button>
</div>
<div id="container">

@Html.EJS().TreeMap("container").AllowImageExport(true).Load("load").WeightV
aluePath("GDP").LeafItemSettings(leaf =>
leaf.LabelPath("State").LabelFormat("${State}
${GDP}
Trillion
(${percentage} %)").LabelStyle(new TreeMapFont { Color =
"#000000" }).Border(new TreeMapBorder { Color = "#000000", Width = 0.5
})).Render()
</div>
<script>
function load(args)
{
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1
 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5
 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
}

```

```

 args.treemap.dataSource = data;
 document.getElementById('togglebtn').onclick = () => {
 args.treemap.export('JPEG', 'TreeMap', null, false).then((data)
=> {
 let base64 = data;
 document.writeln(base64);
 });
 });
 }
}
</script>

```

## EXPORT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

## PDF Export

To use the PDF export functionality, we should set the [allowPdfExport](#) property to **true**. The rendered treemap can be exported as PDF using the [export](#) method. The [export](#) method requires three parameters: file type, file name and orientation of the PDF document. The orientation setting is optional and **0** indicates portrait and **1** indicates landscape.

## CSHTML

```

@using Syncfusion.EJ2;">
<div>
 <button id="togglebtn">Export</button>
</div>
<div id="container">

@Html.EJS().TreeMap("container").Load("load").AllowPdfExport(true).WeightValuePath("GDP").LeafItemSettings(leaf =>
leaf.LabelPath("State").LabelFormat("${State}
${GDP} Trillion
({percentage} %)").LabelStyle(new TreeMapFont { Color = "#000000" }).Border(new TreeMapBorder { Color = "#000000", Width = 0.5 })).Render()
</div>
<script>
 function load(args)

```

```

{
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 document.getElementById('togglebtn').onclick = () => {
 args.treemap.export('PDF', 'TreeMap', 0);
 };
}
</script>

```

### EXPORT.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

**Note:** The exporting of the treemap as base64 string is not supported in the PDF export.

### Accessibility in ASP.NET MVC TreeMap component

TreeMap has built-in accessibility features like screen reading and WAI-ARIA attributes.

#### WAI-ARIA attributes

The TreeMap component follows the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the TreeMap component:

| Attributes | Purpose |

| --- | --- |

| **role=region** | It specifies the TreeMap areas that do not support interactive functions like selection and highlight. |

| **role=button** | It specifies the TreeMap areas where interactive functions such as selection and highlight are available. |

| **aria-label** | Provides an accessible name for the data labels, legend title, and legend item labels. |

### Screen reading in TreeMap

Accessibility in the TreeMap component ensures that all users, regardless of ability or disability, can use screen reading. The following TreeMap elements will be read aloud using screen reading software, such as Narrator for Windows.

| Elements | Description |

| --- | --- |

| Data labels | Reads the labels displayed on leaf items of the TreeMap. |

| Legend title | Reads the title of the legend in the TreeMap. |

| Legend item label | Reads the label of the legend item in the TreeMap. |

### Ensuring accessibility

The TreeMap component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TreeMap component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TreeMap component with accessibility tools.

See also

- [Accessibility in ASP.NET MVC components](#)

### Internationalization

The TreeMap control supports internationalization for the following elements:

- Data label
- Tooltip

For more information about number and date formatter, refer to [internationalization](#).

<!-- markdownlint-disable MD036 -->

### Globalization

Globalization is the process of designing and developing a component that works in different cultures/locales. Internationalization library is used to globalize number, date, and time values in the tree map control using the **format** property in the Treemap.

### Numeric format

In the following code example, tooltip is globalized to Deutsch culture.

### CSHTML

```
@using Syncfusion.EJ2;
```



```

<div id="container">
@Html.EJS().TreeMap("container").Load("load").Format("c").WeightValuePath("GDP").Render()
</div>
<script>
 function load(args)
 {
 var data = [
 { State: "United States", GDP: 17946, percentage: 11.08, Rank: 1 },
 { State: "China", GDP: 10866, percentage: 28.42, Rank: 2 },
 { State: "Japan", GDP: 4123, percentage: -30.78, Rank: 3 },
 { State: "Germany", GDP: 3355, percentage: -5.19, Rank: 4 },
 { State: "United Kingdom", GDP: 2848, percentage: 8.28, Rank: 5 },
 { State: "France", GDP: 2421, percentage: -9.69, Rank: 6 },
 { State: "India", GDP: 2073, percentage: 13.65, Rank: 7 },
 { State: "Italy", GDP: 1814, percentage: -12.45, Rank: 8 },
 { State: "Brazil", GDP: 1774, percentage: -27.88, Rank: 9 },
 { State: "Canada", GDP: 1550, percentage: -15.02, Rank: 10 }
];
 args.treemap.dataSource = data;
 }
</script>

```

## INTERNATIONALIZATION.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 public IActionResult Index()
 {
 return View();
 }
 }
}

```

## Migration from Essential JS 1

This article describes the API migration process of Accordion component from Essential JS 1 to Essential JS 2.

### Data Binding

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| DataSource | **Property:** *dataSource*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>DataSource(datasource)<br/>.WeightValuePath("population"))<br/>&#160;@section ControlsSection{@{ var datasource = ViewData["medals"];}<br/><br/>**script:**<br/><br/>public ActionResult Customization()<br/>{ViewData["medals"] = MedalData.GetPopulation();<br/>return View();}public class MedalData{<br/>public static List<MedalData> GetPopulation()<br/>List<MedalData> medals = new List<MedalData>();<br/>medals.Add(new MedalData(){<br/>Continent: "Asia", Population: 1749046000} return medals; }}| **Property:** *dataSource*<br/>@Html.EJS().TreeMap("container")<br/>.Load("load")<br/>.LayoutType(LayoutMode.Squarified).Render();<br/>function load(args)<br/>{var data=[{ Continent: "Asia", Population: 1749046000}];<br/>args.treemap.dataSource = data;}|

### Appearance

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Layout | **Property:** *itemsLayoutMode*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>ItemsLayoutMode("SliceAndDiceAuto"))| **Property:** *layoutType*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.LayoutType(LayoutMode.Squarified)<br/>.Render();|

| Weight Value Path | **Property:** *weightValuePath*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>.WeightValuePath("Population"))| **Property:** *weightValuePath*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.WeightValuePath("Population")<br/>.Render();|

| Range Color Value Path | **Property:** *colorValuePath*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>.ColorValuePath("Continent"))| **Property:** *rangeColorValuePath*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.ColorValuePath("Continent")<br/>.Render();|

| Equal Color Value Path | Not Applicable | **Property:** *equalColorValuePath*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.equalColorValuePath('Asia').Render();|

| Height | **Property:** *height*<br/><br/>  
 @(Html.EJ().TreeMap("container").Height(50))| **Property:** *height*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Height('50px')<br/>.Render();|

| Width | **Property:** *width*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>Width(400))| **Property:** *width*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Width('400px')<br/>.Render();|

| Theme | Not Applicable | **Property:** *theme*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.theme('Highcontrast').Render();|

| Localization | **Property:** *locale*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>Locale("en-US"))| **Property:** *locale*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Locale("en-US")<br/>.Render();|

| Palette Colors | **Property:** *paletteColorMapping.colors*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>.TreeMapPaletteColorMapping(<br/>colors(['red','gree

n'] ) | **Property:** *palette*  
 @Html.EJS().TreeMap("container")<br/>Palette(new  
 string[]<br/>{'#C33764', '#AB3566'})<br/>.Render(); |  
 | Margin | Not Applicable | **Property:** *margin*  
 @Html.EJS().TreeMap("container")<br/>Margin(mar=>mar.<br/>left(10).top('10'))<br/>.Render(  
 ); |  
 | Resize | **Property:** *enableResize*  
 @(Html.EJ().TreeMap("container").<br/>enableResize(true) | Not Applicable |  
 | Responsive | **Property:**  
 isResponsive<br/>@(Html.EJ().TreeMap("container").<br/>isResponsive(true)) | Not Applicable |

### Leaf Items

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Border Color | **Property:**  
*leafItemSettings.borderBrush*  
 @(Html.EJ().TreeMap("container")<br/>.LeafItemsSetting(I  
 ls => { Ils.showLabels(true)<br/>.borderBrush("blue")<br/>}) | **Property:**  
*leafItemSettings.border*  
 @Html.EJS().TreeMap("container")<br/>.LeafItemSettings(<br/>leaf => leaf.Border(new  
 TreeMapBorder<br/>{ Color = "white", Width = 0.5 })<br/><br/>.Render(); |

| Border Width | **Property:**  
*leafItemSettings.borderThickness*  
 @(Html.EJ().TreeMap("container").<br/>.LeafItemsSett  
 ing(Ils => { Ils.showLabels(true) .borderThickness(5)<br/>}) | **Property:**  
*leafItemSettings.border*  
 @Html.EJS().TreeMap("container")<br/>.LeafItemSettings(<br/>leaf => leaf.Border(new  
 TreeMapBorder<br/>{ Width = 0.5 })<br/><br/>.Render(); |

| Gap Value | **Property:** *leafItemSettings.gap*  
 @(Html.EJ().TreeMap("container").<br/>.LeafItemsSetting(Ils => {  
 Ils.showLabels(true)<br/>.gap(5)<br/>}) | **Property:** *leafItemSettings.gap*  
 @Html.EJS().TreeMap("container")<br/>.LeafItemSettings(<br/>leaf  
 =><br/>leaf.Gap(5)<br/><br/>.Render(); |

| Leaf Item Label | **Property:** *leafItemSettings.itemTemplate*  
 @(Html.EJ().TreeMap("container")<br/>.LeafItemsSetting(Ils  
 => { Ils.showLabels(true)<br/>.ItemTemplate("template")<br/>}) | **Property:**  
*leafItemSettings.labelTemplate*  
 @Html.EJS().TreeMap("container")<br/>.LeafItemSettings(<br  
 />leaf =><br/>leaf.LabelTemplate('template')<br/>.Render(); |

| Leaf Label Path | **Property:** *leafItemSettings.labelPath*  
 @(Html.EJ().TreeMap("container").<br/>.LeafItemsSetting(Ils => {  
 Ils.showLabels(true)<br/>.LabelPath("GameName")<br/>}) | **Property:**  
*leafItemSettings.labelPath*  
 @Html.EJS().TreeMap("container")<br/>.LeafItemSettings(<br/>leaf  
 =><br/>.LabelPath('GameName') ).Render(); |

| Leaf Label Position | **Property:**

`leafItemSettings.labelPosition`  
 @Html.EJ().TreeMap("container").LeafItemsSetting  
 (lls => { lls.showLabels( true) } ) | **Property:**

`leafItemSettings.labelPosition`  
 @Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.LabelPosition('Center').Render();

| Leaf Label Color | Not Applicable | **Property:** `leafItemSettings.fill`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.Fill('red').Render();

| Random Colors | Not Applicable | **Property:** `leafItemSettings.autoFill`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.AutoFill(true).Render();

| Format | Not Applicable | **Property:** `leafItemSettings.labelFormat`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.LabelFormat('\${Continent}-\${Population}').Render();

| Labels Visibility | **Property:**

`leafItemSettings.showLabels`  
 @Html.EJ().TreeMap("container").LeafItemsSet  
 ting(lls => { lls.ShowLabels(true) } ) | **Property:** `leafItemSettings.showLabels`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.ShowLabels(false).Render();

| Opacity | Not Applicable | **Property:** `leafItemSettings.opacity`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.Opacity(0.7).Render();

| Padding | Not Applicable | **Property:** `leafItemSettings.padding`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.Padding(5).Render();

| Font Customization | Not Applicable | **Property:** `leafItemSettings.labelStyle`

@Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.LabelStyle(new TreeMapFont { size= '12px', color= 'red', opacity= 0.5 } )  
 .Render();

| Position of Template | Not Applicable | **Property:**

`leafItemSettings.templatePosition`  
 @Html.EJS().TreeMap("container").LeafItemSettings  
 => leaf.TemplatePosition('Center').Render();

## Legend

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Legend Alignment | **Property:** `legendSettings.alignment`

@Html.EJ().TreeMap("container").TreeMapLegend(t1=> { t1.Alignment("far") } ) | **Property:** `legendSettings.alignment`

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>
t1.Alignment("far")
).Render();|
```

| Legend Visibility | **Property:** *showLegend*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=>{t1.ShowLegend(false)
})| Pro
perty: legendSettings.visible


```

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>t1.ShowLegend(false)
).R
ender();|
```

| Legend Position | **Property:** *legendSettings.dockPosition*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=>{t1.DockPosition("bottom")

})| Property: legendSettings.position


```

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>
t1.position('Top')
).Render();|
```

| Legend Height | **Property:** *legendSettings.height*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=>{t1.Height(100)
})| Property:
legendSettings.height


```

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>
t1.Height('100px')
).Render();|
```

| Legend Width | **Property:** *legendSettings.width*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=>{t1.Width(100)
})| Property:
legendSettings.width


```

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>
t1.Width('100')
).Render();|
```

| Shape Height | **Property:**

```
legendSettings.iconHeight

@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=
>{t1.IconHeight(15)
})| Property:
```

```
legendSettings.shapeHeight

@Html.EJS().TreeMap("container")
.LegendSettings(
t1=> t1.ShapeHeight('40px')
).Render();|
```

| Shape Width | **Property:** *legendSettings.iconWidth*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=>{t1.IconWidth(8)
})| Property:
legendSettings.shapeWidth


```

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>t1.ShapeWidth('40px')
).
Render();|
```

| Padding | Not Applicable | **Property:** *legendSettings.shapePadding*<br/><br/>

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=> t1.ShapePadding
(10)
).Render();|
```

| Legend Title | **Property:** *legendSettings.title*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.TreeMapLegend(t1=>{t1.Title("Population")
})| Pro
perty: legendSettings.title


```

```
@Html.EJS().TreeMap("container")
.LegendSettings(
t1=>t1.Title('Legend')
).Rend
er();|
```

| Legend Shape | Not Applicable | **Property:** *legendSettings.shape*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 t1=>t1.Shape("Rectangle").Render();|

| Legend Mode | **Property:** *legendSettings.mode*  
 @Html.EJS().TreeMap("container").TreeMapLegend(t1=>t1.  
 mode("interactive")) | **Property:** *legendSettings.mode*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 t1=>t1.mode("interactive").Render();|

| Legend Text Customization | Not Applicable | **Property:** *legendSettings.textStyle*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 legend=>legend.TextStyle(new  
 TreeMapFont{ Size= "10px", Opacity= 0.5, Color= "red"}).Render();|

| Legend Title Customization | Not Applicable | **Property:** *legendSettings.titleStyle*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 legend=>legend.TitleStyle(n  
 ew TreeMapFont{ Size= "10px", Opacity= 0.5, Color= "red" }).Render();|

| Legend Shape Border | Not Applicable | **Property:** *legendSettings.shapeBorder*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 legend.ShapeBorder(new  
 TreeMapBorder{ Color= "red", Width= 1 }).Render();|

| Legend Template | **Property:** *legendSettings.template*  
 @Html.EJS().TreeMap("container").TreeMapLegend(t1=>{  
 t1.template("template")}) | Not Applicable |

| Left Label | **Property:** *legendSettings.leftLabel*  
 @Html.EJS().TreeMap("container").TreeMapLegend(t1=>t1.Mode("interactive").Left  
 Label("10Million")) | Not Applicable |

| Right Label | **Property:** *legendSettings.rightLabel*  
 @Html.EJS().TreeMap("container").TreeMapLegend(t1=>t1.Mode("interactive").Left  
 Label("10Million")) | Not Applicable |

| Legend Shape Image | Not Applicable | **Property:** *legendSettings.imageUrl*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 t1=>t1.ImageUrl("")).Render(  
 );|

| Position in Intractive Legend | Not Applicable | **Property:** *legendSettings.labelPosition*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 t1=>t1.LabelPosition("Center")</br>).Render();|

| Legend Location | Not Applicable | **Property:** *legendSettings.location*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 t1=>t1.Location({ x=10, y= 20  
 }).Render();|

| Legend Orientation | Not Applicable | **Property:** *legendSettings.orientation*  
 @Html.EJS().TreeMap("container").LegendSettings(  
 t1=>t1.Orientation("Horizontal")</br>).Render();|

## Levels

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Random Colors | Not Applicable | **Property:** *levels.autoFill*<br/><br/>

```
@(Html.EJS().TreeMap("container")
.Levels(
l1=>l1.AutoFill(true)
).Render();|
```

| Level Background Color | **Property:** *levels.groupBackground*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.Levels(l1=>
{l1.GroupBackground("white")
})|
```

```
Property: levels.fill

 @Html.EJS().TreeMap("container")
.Levels(
l1=>l1.Fill('white')
).Render();|
```

| Level Border Color | **Property:**

```
levels.groupBorderColor

@(Html.EJ().TreeMap("container")
.Levels(l1=>
l1.GroupBorderColor("#58585B") {
})| Property:
```

```
levels.border.color
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.Border(new TreeMapBorder{color = 'black'})
).Render();|
```

| Level Border Width | **Property:** *levels.groupBorderThickness*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.Levels(l1=>
l1.GroupBorderThickness(2)
})|
```

```
Property: levels.border.width


```

```
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.Border(new TreeMapBorder{width =0.5})
).Render();|
```

| Group Gap | **Property:**

```
levels.groupGap

@(Html.EJ().TreeMap("container").
.Levels(l1=>
l1.groupGap(2)
})| Property: levels.groupGap


```

```
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.GroupGap(2)
).Render();|
```

| Group Padding | **Property:**

```
levels.groupPadding

@(Html.EJ().TreeMap("container").
.Levels(l1=>
l1.GroupPadding(1){
})| Property:
```

```
levels.groupPadding
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.GroupPadding(1)
).Render();|
```

| Group Path | **Property:** *levels.groupPath*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.Levels(l1=>
l1.GroupPath("pathname")
})|
```

```
Property: levels.groupPath


```

```
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.GroupPath("pathname")
).Render();|
```

| Height of Header Level | **Property:** *levels.headerHeight*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.Levels(l1=>
l1.HeaderHeight(20)
})| Property:
```

```
levels.headerHeight


```

```
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.HeaderHeight(20)
).Render();|
```

| Header Template | **Property:** *levels.headerTemplate*<br/><br/>

```
@(Html.EJ().TreeMap("container").
.Levels(l1=>
l1.HeaderTemplate("template")
})|
```

```
Property: levels.headerTemplate


```

```
@Html.EJS().TreeMap("container")
.Levels(
l1=>l1.HeaderTemplate("template")
).Render();|
```



| Opacity of Color | Not Applicable | **Property:** *levels.opacity*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Levels(<br/>l1=>l1.Opacity(0.5)<br/>).Render();|

| Header Visibility | **Property:** *levels.showHeader*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>.Levels(l1=><br/>{l1.ShowHeader(false)<br/>}))|  
**Property:** *levels.showHeader*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Levels(<br/>l1=>l1.ShowHeader(false)<br/>).Render();|

| Template Position | **Property:**  
*levels.labelPosition*<br/><br/>@(Html.EJ().TreeMap("container").<br/>.Levels(l1=><br/>{l1.labelPosition("topleft")<br/>}))| **Property:** *levels.templatePosition*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Levels(<br/>l1=>l1.TemplatePosition('Center')<br/>).Render();|

| Header Style | Not Applicable | **Property:** *levels.headerStyle*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Levels(<br/>l1=>l1.HeaderStyle(new TreeMapFont{<br/>Color= 'red', Size= '16px', Opacity= 0.7 })<br/>).Render();|

| Header Format | Not Applicable | **Property:** *levels.headerFormat*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Levels(<br/>l1=>l1.HeaderFormat('\${Continent}')<br/>).Render();|

| Header Alignment | Not Applicable | **Property:** *levels.headerAlignment*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.Levels(<br/>l1=>l1.HeaderAlignment('Center')<br/>).Render();|

## Selection

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Selection | **Property:** *selectionMode*<br/><br/>  
 @(Html.EJ().TreeMap("container").<br/>.selectionMode("default"))| **Property:**  
*selectionSettings.mode*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.SelectionSettings(<br/>selectionSettings=><br/>selectionSettings.Enable(true).Mode('Item')<br/>).Render();|

| Selection Color | Not Applicable | **Property:** *selectionSettings.fill*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.SelectionSettings(<br/>selectionSettings=><br/>selectionSettings.Enable(true).Fill('blue')<br/>).Render();|

| Selection Color Opacity | Not Applicable | **Property:** *selectionSettings.opacity*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.SelectionSettings(<br/>selectionSettings=><br/>selectionSettings.Enable(true)<br/>.Opacity('0.5')<br/>).Render();|

| Border for selection | Not Applicable | **Property:** *selectionSettings.border*<br/><br/>  
 @Html.EJS().TreeMap("container")<br/>.SelectionSettings(<br/>selectionSettings=><br/>selectionSettings.Enable(true).Border(new TreeMapBorder{<br/>Color= 'red', Width=2}<br/>).Render();|

## Hightlight

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |



| --- | --- | --- |

| Highlight Group Selection Mode | **Property:** *highlightGroupOnSelection*<br/><br/>  
@Html.EJ().TreeMap("container")<br/>.highlightGroupOnSelection(true)) | **Property:**  
*highlightSettings.mode*<br/><br/>

@Html.EJS().TreeMap("container")<br/>HighlightSettings<br/>(highlight  
=><br/>highlight.Enable(true)<br/>.Mode('All')<br/>.Render());|

| Highlight Selection Mode | **Property:** *highlightOnSelection*<br/><br/>  
@Html.EJ().TreeMap("container")<br/>.highlightOnSelection(true) | **Property:**  
*highlightSettings.mode*<br/><br/>

@Html.EJS().TreeMap("container")<br/>HighlightSettings<br/>(highlight  
=><br/>highlight.Enable(true)<br/>Mode('Item')<br/>.Render());|

| Highlight Group Border Color | **Property:** *highlightGroupBorderBrush*<br/><br/>  
@Html.EJ().TreeMap("container")<br/>.highlightGroupBorderBrush('gray') | **Property:**  
*highlightSettings.border.color*<br/><br/>

@Html.EJS().TreeMap("container")<br/>HighlightSettings<br/>(highlight  
=><br/>highlight.Enable(true)<br/>.Mode('All')<br/>.Border(new  
TreeMapBorder<br/>{color="red"}).Render());|

| Highlight Group Border Width | **Property:** *highlightGroupBorderThickness*<br/><br/>  
@Html.EJ().TreeMap("container")<br/>.highlightGroupBorderThickness(3) | **Property:**  
*highlightSettings.border.width*<br/><br/>

@Html.EJS().TreeMap("container")<br/>HighlightSettings<br/>(highlight  
=><br/>highlight.Enable(true)<br/>.Mode('All')<br/>.Border(new  
TreeMapBorder<br/>{Width=0.8}).Render());|

| Highlight Color | Not Applicable | **Property:** *highlightSettings.fill*<br/><br/>

@Html.EJS().TreeMap("container")<br/>HighlightSettings<br/>(highlight  
=><br/>highlight.Enable(true)<br/>.Fill('red')<br/>.Render());|

| Highlight Color Opacity | Not Applicable | **Property:** *highlightSettings.opacity*<br/><br/>

@Html.EJS().TreeMap("container")<br/>HighlightSettings<br/>(highlight  
=><br/>highlight.Enable(true)<br/>Fill('red')<br/>.Opacity(0.5)).Render());|

### Range ColorMapping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| From value | **Property:**

*rangeColorMapping.from*<br/><br/>@Html.EJ().TreeMap("container")<br/>.TreeMapRangeColor  
Mappings<br/>cm =>cm.From(1000)<br/>)) | **Property:**  
*leafItemSettings.colorMapping.from*<br/><br/>

@Html.EJS().TreeMap("container")<br/>.Load("load").Render();<br/>function  
load(args)<br/>{args.treemap.leafItemSettings<br/>.colorMapping[0]  
=<br/>{<br/>from:1000}<br/>}}|

| To value | **Property:** *rangeColorMapping.to*<br/><br/>

@Html.EJ().TreeMap("container")<br/>.TreeMapRangeColorMappings<br/>cm

```

=>cm.To(1000)
)| Property: leafItemSettings.colorMapping.to

@Html.EJS().TreeMap("container")
.Load("load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
to:10000}
}|

|Color| Property: rangeColorMapping.color

@Html.EJ().TreeMap("container")
.TreeMapRangeColorMappings(
cm
=>cm.Color('red')
)| Property: leafItemSettings.colorMapping.color

@Html.EJS().TreeMap("container")
.Load("load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
color:'red'}
}|

|Legend Label| Property: rangeColorMapping.legendLabel

@Html.EJ().TreeMap("container")
.TreeMapRangeColorMappings(
cm
=>cm.LegendLabel("Growth")
)| Property:
leafItemSettings.colorMapping.label
@Html.EJS().TreeMap("container")
.Load("load").Re
nder();
function load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
label: "Growth"}
}|

```

#### Desaturation ColorMapping

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

```

|From value| Property: desaturationColorMapping.from

@Html.EJ().TreeMap("container")
.TreeMapDesaturationColorMapping(
cm
=>
cm.From(1))| Property: leafItemSettings.colorMapping.from

@Html.EJS().TreeMap("container")
.Load("load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
from:1000
}}|

```

```

|To value| Property: desaturationColorMapping.to

@Html.EJ().TreeMap("container")
.TreeMapDesaturationColorMapping(
cm
=>
cm.To(0.2))| Property: leafItemSettings.colorMapping.to

@Html.EJS().TreeMap("container")
.Load("load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
to:10000
}}|

```

```

|Color| Property: desaturationColorMapping.color

@Html.EJ().TreeMap("container")
.TreeMapDesaturationColorMapping(
cm
=>
cm.color('red'))| Property: leafItemSettings.colorMapping.color

@Html.EJS().TreeMap("container")
.Load("load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
color:'red'}
}}|

```

```

|Value| Not Applicable| Property:
leafItemSettings.colorMapping.value
@Html.EJS().TreeMap("container")
.Load("load").Re
nder();
function load(args)
{args.treemap.leafItemSettings
.colorMapping[0]
=
{
value: "Population"
}}|

```

| Minimum Opacity | Not Applicable | **Property:**

```
leafItemSettings.colorMapping.minOpacity

@Html.EJS().TreeMap("container")
.Load(
"load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0] =
{
minOpacity:
0.7
}}|
```

| Maximum Opacity | Not Applicable | **Property:**

```
leafItemSettings.colorMapping.maxOpacity

@Html.EJS().TreeMap("container")
.Load
("load").Render();
function
load(args)
{args.treemap.leafItemSettings
.colorMapping[0] =
{
maxOpacity:
0.7}
}}|
```

### Tooltip

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Tooltip | **Property:** *showTooltip*<br/>@Html.EJ().TreeMap("container")<br/>.ShowTooltip(true)) |  
**Property:** *tooltipSettings.visible*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new TreeMapTooltipSettings{Visible  
=true}<br/>).Render();|

| Tooltip Template | **Property:** *tooltipTemplate*<br/><br/>  
@Html.EJ().TreeMap("container")<br/>.ShowTooltip(true)<br/>.TooltipTemplate("template")) |  
**Property:** *tooltipSettings.template*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new  
TreeMapTooltipSettings{<br/>Visible =true,<br/>template='template'}<br/>).Render();|

| Tooltip Border | Not Applicable | **Property:** *tooltipSettings.border*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new  
TreeMapTooltipSettings{<br/>Visible=true,<br/>Border=(new TreemapBorder{Color='red'})  
}<br/>).Render();|

| Tooltip Color | Not Applicable | **Property:** *tooltipSettings.fill*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new  
TreeMapTooltipSettings{<br/>Visible =true , Fill='red'}<br/>).Render();|

| Tooltip Format | Not Applicable | **Property:** *tooltipSettings.format*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new  
TreeMapTooltipSettings{<br/>Visible =true, Format='\${Population}'}<br/>).Render();|

| Tooltip Marker Shape | Not Applicable | **Property:** *tooltipSettings.markerShapes*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new  
TreeMapTooltipSettings{<br/>Visible =true, MarkerShapes='Circle' }<br/>).Render();|

| Tooltip Color Opacity | Not Applicable | **Property:** *tooltipSettings.opacity*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new  
TreeMapTooltipSettings{<br/>Visible =true, Opacity: 0.5 }<br/>).Render();|

| Tooltip Text Style | Not Applicable | **Property:** *tooltipSettings.textStyle*<br/><br/>  
@Html.EJS().TreeMap("container")<br/>.TooltipSettings(new TreeMapTooltipSettings

```
{
Visible =true
TextStyle=(new TreeMapFont { Color= 'red', Opacity=0.5, Size= '12px'
}}).Render();|
```

### Drilldown

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Drilldown | **Property:**

*enableDrillDown*<br/>@(Html.EJ().TreeMap("container")<br/>.EnableDrillDown(true)) | **Property:**  
*enableDrillDown*<br/><br/>

@Html.EJS().TreeMap("container")<br/>.EnableDrillDown(true).Render();|

| Drilldown Level | **Property:**

*drillDownLevel*<br/>@(Html.EJ().TreeMap("container")<br/>.DrillDownLevel(1)) | **Property:**

*InitialDrillSettings.groupIndex*<br/><br/>

@Html.EJS().TreeMap("container")<br/>.EnableDrillDown(true)<br/>InitialDrillDown<br/>(new  
TreeMapInitialDrillSettings<br/>{ GroupIndex =1}).Render();|

### Methods

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| Treemap Refresh Method | **Method:** *refresh*<br/><br/> var treemap = <br/> &#160;

\$("#container").ejTreeMap("instance"); <br/> treemap.refresh();| **Method:** *refresh*<br/><br/> var  
treemap = <br/> &#160; document.getElementById('container').ej2\_instances[0]; <br/>  
treemap.refresh();|

| Method to Drilldown | **Method:** *drillDown*<br/><br/> var treemap = <br/> &#160;

\$("#container").ejTreeMap("instance"); <br/> treemap.drillDown();| Not Applicable|

| Append to Method | Not Applicable | **Method:** *appendTo*<br/><br/> var treemap = <br/> &#160;

document.getElementById('container').ej2\_instances[0]; <br/> treemap.appendTo();|

| Add Event Listener Method | Not Applicable | **Method:** *addEventListener*<br/><br/> var treemap =

<br/> &#160; document.getElementById('container').ej2\_instances[0]; <br/>

treemap.addEventListener();|

| Treemap Destroy Method | Not Applicable | **Method:** *destroy*<br/><br/> var treemap = <br/> &#160;

document.getElementById('container').ej2\_instances[0]; <br/> treemap.destroy();|

| Treemap Exporting Method | Not Applicable | **Method:** *export*<br/><br/> var treemap = <br/> &#160;

document.getElementById('container').ej2\_instances[0]; <br/> treemap.export();|

| Get the Module Name | Not Applicable | **Method:** *getModuleName*<br/><br/> var treemap = <br/>

&#160; document.getElementById('container').ej2\_instances[0]; <br/>

treemap.getModuleName();|

| Printing the Treemap | Not Applicable | **Method:** *print*<br/><br/> var treemap = <br/> &#160;

document.getElementById('container').ej2\_instances[0]; <br/> treemap.print();|

| Resizing the TreeMap | Not Applicable | **Method:** *resizeOnTreeMap*  
 &#160; document.getElementById('container').ej2\_instances[0];  
 treemap.resizeOnTreeMap(); |

| Inject Method (Tooltip) | Not Applicable | **Method:** *resizeOnTreeMap*  
 TreeMap.Inject(TreeMapTooltip);  
 document.getElementById('container').ej2\_instances[0];  
 treemap.resizeOnTreeMap(); |

| Remove Event Listener Method | Not Applicable | **Method:** *removeEventListener*  
 treemap = &#160; document.getElementById('container').ej2\_instances[0];  
 treemap.removeEventListener(); |

## Events

| **Behavior** | **API in Essential JS 1** | **API in Essential JS 2** |

| --- | --- | --- |

| TreeMap Load Event | Not Applicable | **Event:** *load*  
 @Html.EJS().TreeMap("container").Load("load").Render();  
 itemMove(args); |

| TreeMap Loaded Event | Not Applicable | **Event:** *loaded*  
 @Html.EJS().TreeMap("container").Loaded("load").Render();  
 load(args); |

| Event Before Print | Not Applicable | **Event:**  
*beforePrint*  
 @Html.EJS().TreeMap("container").BeforePrint("beforePrint").Render();  
 beforePrint(args); |

| Click Event | **Event:**  
*click*  
 @Html.EJS().TreeMap("container").Click("click");  
 click(args); | **Event:** *click*  
 @Html.EJS().TreeMap("container").Click("click").Render();  
 click(args); |

| Drill Start Event | **Event:**  
*drillStarted*  
 @Html.EJS().TreeMap("container").DrillStarted("drillStarted");  
 drillStarted(args); | **Event:** *drillStart*  
 @Html.EJS().TreeMap("container").DrillStart("drillStart").Render();  
 drillStart(args); |

| Drill End Event | Not Applicable | **Event:** *drillEnd*  
 @Html.EJS().TreeMap("container").DrillEnd("drillEnd").Render();  
 drillEnd(args); |

| Event on Item Click | Not Applicable | **Event:** *itemClick*  
 @Html.EJS().TreeMap("container").ItemClick("itemClick").Render();  
 itemClick(args); |

| Event Before Print | Not Applicable | **Event:**  
*beforePrint*  
 @Html.EJS().TreeMap("container").BeforePrint("beforePrint").Render();  
 beforePrint(args); |

| Click Event | **Event:**

```
click

@(Html.EJ().TreeMap("container")
.Click("click"))</br><script></br>function
click(args)</br>{</br>}| Event: click

@Html.EJS().TreeMap("container")
.Click("click").Render();
function
click(args)
{
}
```

| Drill Start Event | **Event:**

```
drillStarted

@(Html.EJ().TreeMap("container")
.DrillStarted("drillStarted"))</br>fu
nction drillStarted(args)</br>{| Event: drillStart

@Html.EJS().TreeMap("container")
.DrillStart("drillStart").Render();
function
drillStart(args)
{
}
```

| Drill End Event | Not Applicable | **Event:** drillEnd<br/><br/>

```
@Html.EJS().TreeMap("container")
.DrillEnd("drillEnd").Render();
function
drillEnd(args)
{
}
```

| Event on Item Click | Not Applicable | **Event:** itemClick<br/><br/>

```
@Html.EJS().TreeMap("container").ItemClick("itemClick").Render();
function
itemClick(args)
{
}
```

| Treemap Item Select Event | **Event:** treeMapItemSelected<br/><br/>

```
@(Html.EJ().TreeMap("container")
.TreeMapItemSelected("treeMapItemSelected"))

function treeMapItemSelected(args)</br>{ </br>}| Event: itemSelected

@Html.EJS().TreeMap("container")
.ItemSelected("itemSelected").Render();
function
itemSelected(args)
{
}
```

## How To

### How To

```
<!-- markdownlint-disable MD036 -->
```

[Customize the header for treemap drilldown](#)

```
<!-- markdownlint-disable MD033 -->
```

You can add a header element as <div> and customize it to show the population of a particular country or continent on treemap drill-down.

To customize the header for treemap drill-down, follow the given steps:

#### Step 1:

```
<!-- markdownlint-disable MD031 -->
```

Initialize the treemap and enable the drill-down option.

```
`html
```

```
@Html.EJS().TreeMap("container").WeightValuePath("Population").Format("n").UseGroupingSeparator(
true).WeightValuePath("Population").Palette(new string[] { "#9999ff", "#CCFF99", "#FFFF99",
"#FF9999", "#FF99FF", "#FFCC66" }).LeafItemSettings(leaf =>
```

```
leaf.LabelPath("Name").ShowLabels(false)).EnableDrillDown(true).Levels(level =>
```

```
{
```

```
level.GroupPath("Continent").Fill("#336699").Border(br => br.Color("black").Width(0.5)).Add();
```



```
level.GroupPath("States").Fill("#336699").Border(br => br.Color("black").Width(0.5)).Add();
level.GroupPath("Region").Fill("#336699").Border(br => br.Color("black").Width(0.5)).Add();
}).Render()
`
```

**Step 2:**

Show the population of a particular continent in the treemap **loaded** event. In this event, you can get the header element.

```
`javascript
loaded: function (args: ILoadedEventArgs) {
var header = document.getElementById('header');
var population = 0;
for (var i = 0; i < args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
population += +(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Population);
}
header.innerHTML = 'Continent - Population : ' + population
}
`
```

**Step 3:**

Customize the population for drilled countries or states in the header element when drill-down the treemap. The **drillEnd** event will be triggered when treemap is drilled.

**CSHTML**

```
@using Syncfusion.EJ2;
<div id="header" style="background-color: #179bd7"></div>
<div id="container">

@Html.EJS().TreeMap("container").Load("load").DrillEnd("drillEnd").Loaded("loaded").WeightValuePath("Population").Format("n").UseGroupingSeparator(true).WeightValuePath("Population").Palette(new string[] { "#9999ff", "#CCFF99", "#FFFF99", "#FF9999", "#FF99FF", "#FFCC66" }).LeafItemSettings(leaf =>

leaf.LabelPath("Name").ShowLabels(false)).EnableDrillDown(true).Levels(level =>

{
 level.GroupPath("Continent").Fill("#336699").Border(br =>
br.Color("black").Width(0.5)).Add();
 level.GroupPath("States").Fill("#336699").Border(br =>
br.Color("black").Width(0.5)).Add();
 level.GroupPath("Region").Fill("#336699").Border(br =>
br.Color("black").Width(0.5)).Add();
}).Render()

</div>
<script>
```

```

function loaded(args) {
 var header = document.getElementById('header');
 var population = 0;
 for (var i = 0; i <
args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
 population +=
+(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Populati
on);
 }
 header.innerHTML = 'Continent - Population : ' + population
}
function drillEnd(args) {
 var header = document.getElementById('header');
 var layout =
document.getElementById("container_TreeMap_Squarified_Layout");
 var population = 0;
 if (args.treemap.layout.renderItems[0]['isDrilled']) {
 for (var i = 0; i < args.treemap.layout.renderItems.length; i++)
 {
 population +=
+(args.treemap.layout.renderItems[i]['data'].Population);
 }
 header.innerHTML =
layout.children[0].children[1].innerHTML.split(' ')[1] + ' - ' + population;
 }
 else if (args.treemap.layout.renderItems[0]['parent'].Continent) {
 for (var i = 0; i <
args.treemap.layout.renderItems[0]['parent'].Continent.length; i++) {
 population +=
+(args.treemap.layout.renderItems[0]['parent'].Continent[i]['data'].Populati
on);
 }
 header.innerHTML = 'Continent - Population : ' + population;
 } else {
 population =
args.treemap.layout.renderItems[0]['data'].Population;
 header.innerHTML =
layout.children[0].children[1].innerHTML.split(' ')[1] + ' - Population : '
+ population;
 }
}
function load(args)
{
 var data = [
 { Continent:[
 { Name: "Africa",Population: 1216130000, States: [
 { Name: "Eastern Africa",Population:410637987, Region:[
 { Name:"Ethiopia", Population: 107534882},
 { Name:"Tanzania", Population: 59091392},
 { Name:"Kenya", Population: 50950879},
 { Name:"Uganda", Population: 44270563},
 { Name:"Mozambique", Population: 30528673},
 { Name:"Madagascar", Population: 26262810},
 { Name:"Malawi", Population: 19164728},
 { Name:"Zambia", Population: 17609178},
 { Name:"Zimbabwe", Population: 16913261},
]
 }
]
 }
]
}

```



```

 { Name:"Somalia", Population: 15181925},
 { Name:"South, Sudan", Population: 12919053},
 { Name:"Rwanda", Population: 12501156},
 { Name:"Burundi", Population: 11216450},
 { Name:"Eritrea", Population: 5187948},
 { Name:"Mauritius", Population: 1268315},
 { Name:"Djibouti", Population: 971408},
 { Name:"Réunion", Population: 883247},
 { Name:"Comoros", Population: 832347},
 { Name:"Mayotte", Population: 259682},
 { Name:"Seychelles", Population: 95235},
] },
 { Name: "Middle Africa",Population:158562976, Region:[
 { Name:"Democratic, Republic of the Congo",
Population: 84004989},
 { Name:"Angola", Population: 30774205},
 { Name:"Cameroon", Population: 24678234},
 { Name:"Chad", Population: 15353184},
 { Name:"Congo", Population: 5399895},
 { Name:"Central African, Republic",
Population: 4737423},
 { Name:"Gabon", Population: 2067561},
 { Name:"Equatorial Guinea", Population:
1313894},
 { Name:"Sao Tome and Principe", Population:
208818},
] },
 { Name: "Northern Africa",Population:229385603, Region: [
 { Name:"Egypt", Population: 99375741},
 { Name:"Algeria", Population: 42008054},
 { Name:"Sudan", Population: 41511526},
 { Name:"Morocco", Population: 36191805},
 { Name:"Tunisia", Population: 11659174},
 { Name:"Libya", Population: 6470956},
 { Name:"Western, Sahara", Population:
567421},
] },
 { Name: "Southern Africa",Population:64292365, Region:[
 { Name:"South Africa", Population: 57398421},
 { Name:"Namibia", Population: 2587801},
 { Name:"Botswana", Population: 2333201},
 { Name:"Lesotho", Population: 2263010},
 { Name:"Swaziland", Population: 1391385},
] },
 { Name: "Western Africa", Population:362201579, Region:[
 { Name:"Nigeria", Population: 195875237},
 { Name:"Ghana", Population: 29463643},
 { Name:"Côte d'Ivoire", Population:
24905843},
 { Name:"Niger", Population: 22311375},
 { Name:"Burkina Faso", Population: 19751651},
 { Name:"Mali", Population: 19107706},
 { Name:"Senegal", Population: 16294270},
 { Name:"Guinea", Population: 13052608},
 { Name:"Benin", Population: 11485674},
 { Name:"Togo", Population: 7990926},
 { Name:"Sierra Leone", Population: 7719729},
] },

```

```

 { Name:"Liberia", Population: 4853516},
 { Name:"Mauritania", Population: 4540068},
 { Name:"Gambia", Population: 2163765},
 { Name:"Guinea-Bissau", Population: 1907268},
 { Name:"Cabo Verde", Population: 553335},
 { Name:"Saint Helena", Population: 4074},
] },
],
 },
 { Continent:[
 {
 Name: "Asia", Population:4436224000, States:[
 {Name: "Central Asia", Population: 69787760, Region:[
 { Name:"Uzbekistan", Population: 32364996 },
 { Name:"Kazakhstan", Population: 18403860 },
 { Name:"Tajikistan", Population: 9107211 },
 { Name:"Kyrgyzstan", Population: 6132932 },
 { Name:"Turkmenistan", Population: 5851466 },
] },
 {Name: "Eastern Asia", Population:1641908531,
 Region:[
 { Name:"China", Population: 1415045928},
 { Name:"Japan", Population: 127185332 },
 { Name:"South Korea", Population:
 51164435 },
 { Name:"North Korea", Population:25610672 },
 { Name:"Taiwan", Population: 23694089 },
 { Name:"Hong Kong", Population: 7428887 },
 { Name:"Mongolia", Population: 3121772},
 { Name:"Macao", Population: 632418 },
] },
 {Name: "Southeastern Asia", Population:641775797,
 Region: [
 { Name:"Indonesia", Population:
 266794980 },
 { Name:"Philippines", Population:
 106512074 },
 { Name:"Viet Nam", Population: 96491146 },
 { Name:"Thailand", Population: 69183173 },
 { Name:"Myanmar", Population: 53855735 },
 { Name:"Malaysia", Population: 32042458 },
 { Name:"Cambodia", Population: 16245729 },
 { Name:"Laos", Population: 6961210 },
 { Name:"Singapore", Population:
 5791901},
 { Name:"Timor-Leste", Population:
 1324094},
 { Name:"Brunei Darussalam", Population:
 434076},
] },
 {Name: "Southern Asia", Population: 1846266634,
 Region: [
 { Name:"India", Population: 1354051854},

```

```

 { Name:"Pakistan", Population: 200813818},
 { Name:"Bangladesh", Population:
166368149},

 { Name:"Iran", Population: 82011735},
 { Name:"Afghanistan", Population:
36373176},

 { Name:"Nepal", Population: 29624035},
 { Name:"Sri Lanka", Population:
20950041},

 { Name:"Bhutan", Population: 817054},
 { Name:"Maldives", Population: 444259},
]},
 {Name: "Western Asia", Population:262938009, Region:
[
 { Name:"Turkey", Population: 81916871},
 { Name:"Iraq", Population: 39339753},
 { Name:"Saudi Arabia", Population: 33554343},
 { Name:"Yemen", Population: 28915284},
 { Name:"Syria", Population: 18284407},
 { Name:"Azerbaijan", Population: 9923914},
 { Name:"Jordan", Population: 9903802},
 { Name:"United Arab Emirates", Population:
9541615},

 { Name:"Israel", Population: 8452841},
 { Name:"Lebanon", Population: 6093509},
 { Name:"State of Palestine", Population:
5052776},

 { Name:"Oman", Population: 4829946},
 { Name:"Kuwait", Population: 4197128},
 { Name:"Georgia", Population: 3907131},
 { Name:"Armenia", Population: 2934152},
 { Name:"Qatar", Population: 2694849},
 { Name:"Bahrain", Population: 1566993},
 { Name:"Cyprus", Population: 1189085},
] },
]
 },
 { Continent:[
 {
 Name: "North America", Population:579024000, States:[
 {Name:"Central America", Population:174988756 , Region:[
 { Name:"Mexico", Population: 130759074 },
 { Name:"Guatemala", Population: 17245346 },
 { Name:"Honduras", Population: 9417167 },
 { Name:"El, Salvador", Population: 6411558},
 { Name:"Nicaragua", Population: 6284757 },
 { Name:"Costa, Rica", Population: 4953199},
 { Name:"Panama", Population: 4162618 },
 { Name:"Belize", Population: 382444 },
]},
 { Name:"Northern America", Population:358593810, Region:[
 { Name:"U.S.", Population:3267667480 },
 { Name:"Canada", Population:36953765},

```

```

 { Name:"Bermuda", Population:61070 },
 { Name:"Greenland", Population:56565},
 { Name:"Saint Pierre & Miquelon", Population:6342
 },
]},
]
}]
},

{
 Continent:[
 {
 Name:"South America", Population: 422535000, States:[
 {Name:"Brazil", Population:204519000},
 {Name:"Colombia", Population:48549000 },
 {Name:"Argentina", Population:43132000 },
 {Name:"Peru", Population:31153000 },
 {Name:"Venezuela", Population:30620000},
 {Name:"Chile", Population:18006000 },
 {Name:"Ecuador", Population:16279000},
 {Name:"Bolivia", Population:10520000},
 {Name:"Paraguay", Population:7003000},
 {Name:"Uruguay", Population:3310000},
 {Name:"Guyana", Population:747000},
 {Name:"Suriname", Population:560000 },
 {Name:"French Guiana", Population:262000},
 {Name:"Falkland Islands", Population:3000 },
]},
]},
 { Continent:[
 {
 Name: "Europe", Population:738849000, States:[
 {Name:"Eastern Europe", Population:291953328, Region:[
 {Name:"Russia", Population:143964709 },
 {Name:"Ukraine", Population: 44009214},
 {Name:"Poland", Population:38104832 },
 {Name:"Romania", Population:19580634 },
 {Name:"Czech Republic", Population:10625250 },
 {Name:"Hungary", Population:9688847 },
 {Name:"Belarus", Population:9452113 },
 {Name:"Bulgaria", Population: 7036848},
 {Name:"Slovakia", Population: 5449816},
 {Name:"Moldova", Population:4041065 },
] },
 {Name:"Northern Europe", Population:103642971, Region:[
 { Name:"United Kingdom", Population: 66573504},
 { Name:"Sweden", Population: 9982709},
 { Name:"Denmark", Population: 5754356},
 { Name:"Finland", Population: 5542517},
 { Name:"Norway", Population: 5353363},
 { Name:"Ireland", Population: 4803748},
 { Name:"Lithuania", Population: 2876475},
 { Name:"Latvia", Population: 1929938},
 { Name:"Estonia", Population: 1306788},
 { Name:"Iceland", Population: 337780},
] },
] },
] },
] },
] },
}

```

```

 { Name:"Channel Islands", Population: 166083},
 { Name:"Isle of Man", Population: 84831},
 { Name:"Faeroe Islands", Population: 49489},
] },
 {Name:"Southern Europe", Population:152172107, Region:[
 { Name:"Italy",Population: 59290969 },
 { Name:"Spain",Population: 46397452},
 { Name:"Greece",Population: 11142161 },
 { Name:"Portugal",Population: 10291196},
 { Name:"Serbia",Population: 8762027 },
 { Name:"Croatia",Population: 4164783 },
 { Name:"Bosnia and Herzegovina",Population: 3503554
 },

 { Name:"Albania",Population: 2934363 },
 { Name:"Macedonia",Population: 2085051 },
 { Name:"Slovenia",Population: 2081260 },
 { Name:"Montenegro",Population: 629219},
 { Name:"Malta",Population: 432089 },
 { Name:"Andorra",Population: 76953 },
 { Name:"Gibraltar",Population: 34733 },
 { Name:"San Marino",Population: 33557 },
 { Name:"Holy, See",Population: 801 },
]},
 {Name:"Western Europe", Population:92746859, Region:[
 { Name:"Germany", Population: 82293457 },
 { Name:"France", Population: 65233271 },
 { Name:"Netherlands", Population: 17084459 },
 { Name:"Belgium", Population: 11498519 },
 { Name:"Austria", Population: 8751820 },
 { Name:"Switzerland", Population: 8544034 },
 { Name:"Luxembourg", Population: 590321 },
 { Name:"Monaco", Population: 38897 },
 { Name:"Liechtenstein", Population: 38155 },
]},
]},
]}
];
 args.treemap.dataSource = data;
}
</script>

```

## HEADER.CS

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {

```

```

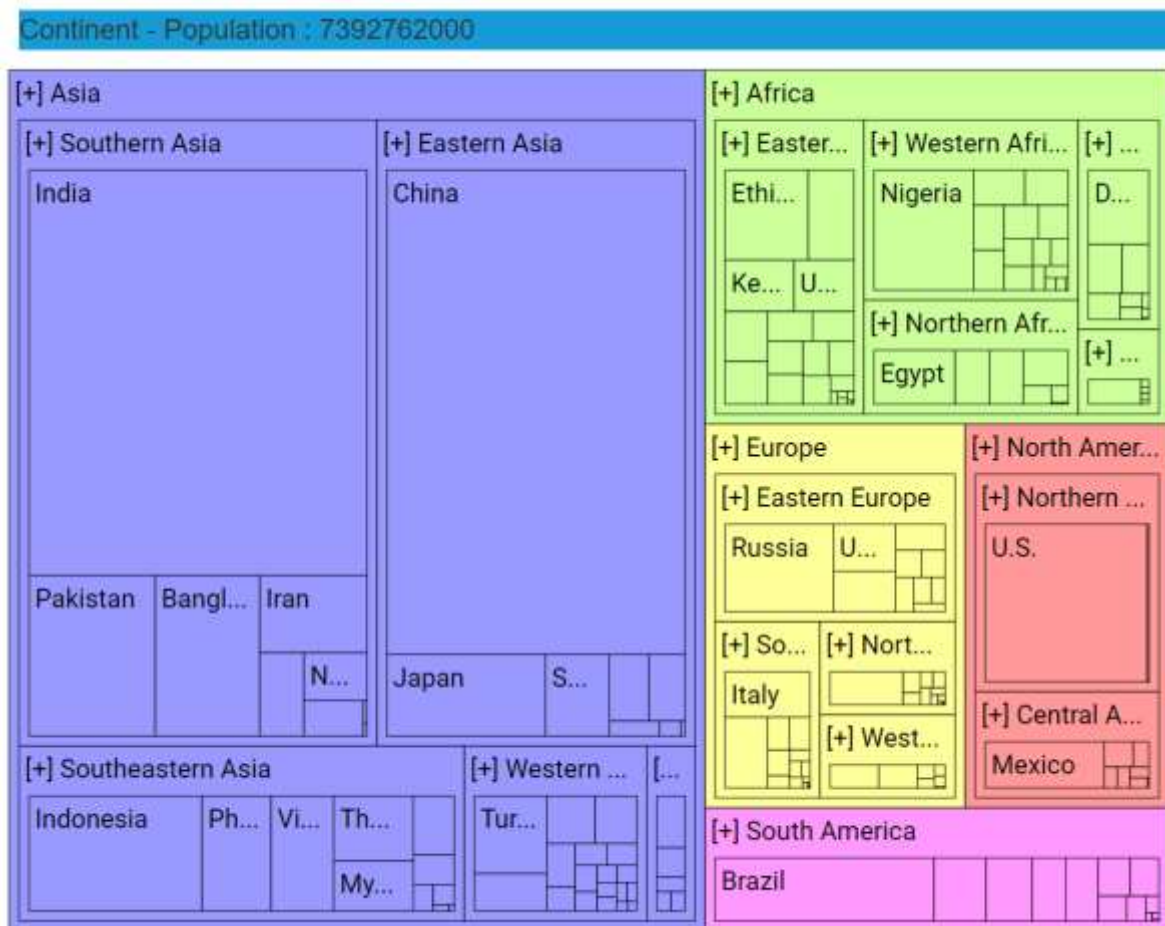
public IActionResult Index()
{
 return View();
}

```

### Sample reference

[treemap sample.](#)

### Screenshot



#### Add label template with drill down

You can add a label template as `<div>` element to the treemap control when using the label template. To add a label template to the treemap control, you have to hide another labels by setting the `showLabels` property to **false** in `leafItemSettings` to show only the label template.

To add label template to treemap drilldown, follow the given steps:

#### Step 1:

Create a treemap control and enable the drill-down option.

```
`html
```

```
@Html.EJS().TreeMap("container").Load("load").DrillStart("drillStart").WeightValuePath("Sales").Palette
(new string[] { "white"}).LeafItemSettings(leaf =>
leaf.ShowLabels(false).LabelTemplate("#template").TemplatePosition(Syncfusion.EJ2.TreeMap.LabelPosi
tion.Center)).EnableDrillDown(true).Levels(level =>
{
level.GroupPath("Continent").Border(br => br.Color("black").Width(0.5)).Add();
level.GroupPath("Company").Border(br => br.Color("black").Width(0.5)).Add();
}).Render()
,
```

### Step 2:

Add the label template in the `leafItemSettings` options, and then set the `showLabels` property to **false** to hide another labels and show only label template.

### CSHTML

```
@using Syncfusion.EJ2;
<div class="row">

@Html.EJS().TreeMap("container").Load("load").DrillStart("drillStart").Weigh
tValuePath("Sales").Palette(new string[] { "white"}).LeafItemSettings(leaf
=>

leaf.ShowLabels(false).LabelTemplate("#template").TemplatePosition(Syncfusio
n.EJ2.TreeMap.LabelPosition.Center)).EnableDrillDown(true).Levels(level =>
{
 level.GroupPath("Continent").Border(br =>
br.Color("black").Width(0.5)).Add();
 level.GroupPath("Company").Border(br =>
br.Color("black").Width(0.5)).Add();
}).Render()

</div>
<div id="template" style="display:none">
 <div style="background-color: red">{{:Company}}</div>
</div>
<script>
 function drillStart(args) {
 var labelElementGroup =
document.getElementById('container_Label_Template_Group');
 labelElementGroup.remove();
 }
 function load(args) {
 var data = [
{ Continent: "China", Company: "Volkswagen", Sales: 3005994 },
{ Continent: "China", Company: "General Motors", Sales: 1230044 },
{ Continent: "China", Company: "Honda", Sales: 1197023 },
{ Continent: "United States", Company: "General Motors", Sales: 3042775 },
{ Continent: "United States", Company: "Ford", Sales: 2599193 },
{ Continent: "United States", Company: "Toyota", Sales: 2449587 },
{ Continent: "Japan", Company: "Toyota", Sales: 1527977 },
{ Continent: "Japan", Company: "Honda", Sales: 706982 },
{ Continent: "Japan", Company: "Suzuki", Sales: 623041 },
```

```

{ Continent: "Germany", Company: "Volkswagen", Sales: 655977 },
{ Continent: "Germany", Company: "Mercedes", Sales: 310845 },
{ Continent: "Germany", Company: "BMW", Sales: 261931 },
{ Continent: "United Kingdom", Company: "Ford ", Sales: 319442 },
{ Continent: "United Kingdom", Company: "Vauxhall", Sales: 251146 },
{ Continent: "United Kingdom", Company: "Volkswagen", Sales: 206994 }
];
 args.treemap.dataSource = data;
}
</script>

```

## LABEL.CS

```

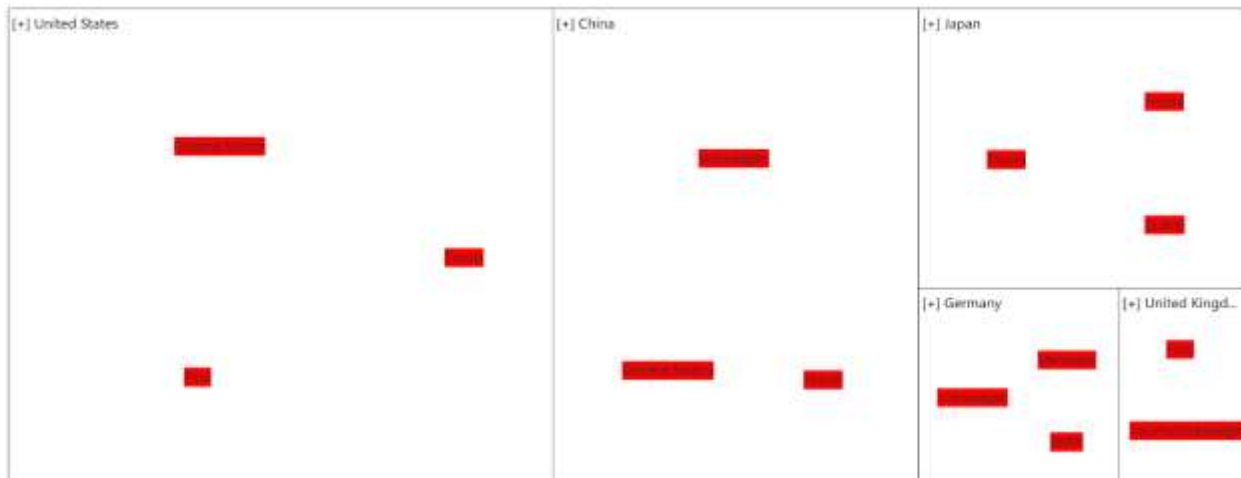
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using EJ2_Core_Application.Models;
using Newtonsoft.Json;
namespace EJ2_Core_Application.Controllers
{
 public class HomeController : Controller
 {
 {
 public IActionResult Index()
 {
 {
 return View();
 }
 }
 }
 }
}

```

## Sample reference

[treemap sample.](#)

## Screenshot





## TreeView

### Getting Started with ASP.NET MVC TreeView Control

This section briefly explains about how to include [ASP.NET MVC TreeView](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

##### [System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

Install ASP.NET MVC package in the application

To add **ASP.NET MVC** controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](https://nuget.org). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5 NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

,

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

,

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

#### **~/ LAYOUT.CSHTML**

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
```

```
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

Register Syncfusion script manager

Also, register the script manager `EJS().ScriptManager()` at the end of `<body>` in the `~/Pages/Shared/_Layout.cshtml` file as follows.

#### ~/ \_LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

Add ASP.NET MVC TreeView control

Now, add the Syncfusion ASP.NET MVC TreeView control in `~/Views/Home/Index.cshtml` page.

#### CSHTML

```
@Html.EJS().TreeView("listdata").Render();
```

Binding data source

TreeView can load data either from local data sources or remote data services. This can be done using the `dataSource` property that is a member of the `fields` property. The `dataSource` property supports array of JavaScript objects and `DataManager`. Here, an array of JSON values is passed to the TreeView control.

#### CSHTML

```
@model List<object>
@Html.EJS().TreeView("listdata").Fields(field=>
 field.Id("id").ParentID("pid").Selected("selected").
 Expanded("expanded").Text("name").HasChildren("hasChild")
 .DataSource(Model)).Render()
```

#### SELFREFERENTIAL.CS

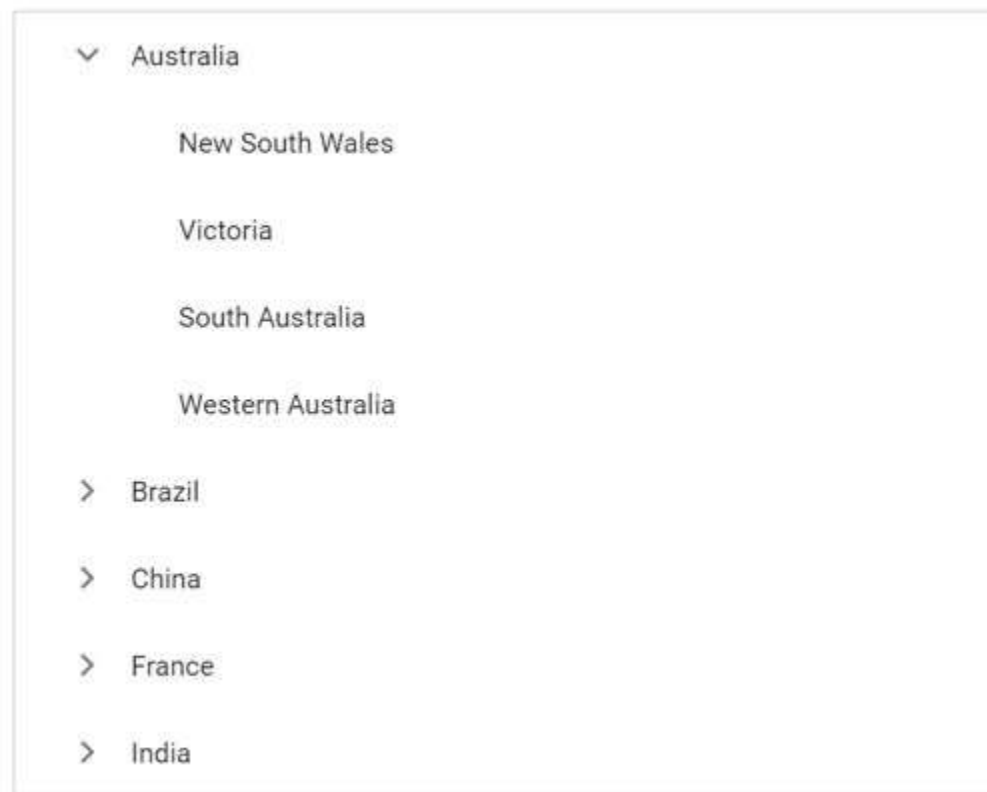
```
public ActionResult Index()
{
 List<object> listdata = new List<object>();
 listdata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
```

```
expanded = true
});
listdata.Add(new
{
 id = 2,
 pid = 1,
 name = "New South Wales",
});
listdata.Add(new
{
 id = 3,
 pid = 1,
 name = "Victoria"
});
listdata.Add(new
{
 id = 4,
 pid = 1,
 name = "South Australia"
});
listdata.Add(new
{
 id = 6,
 pid = 1,
 name = "Western Australia",
});
listdata.Add(new
{
 id = 7,
 name = "Brazil",
 hasChild = true
});
listdata.Add(new
{
 id = 8,
 pid = 7,
 name = "Paraná"
});
listdata.Add(new
{
 id = 9,
 pid = 7,
 name = "Ceará"
});
listdata.Add(new
{
 id = 10,
 pid = 7,
 name = "Acre"
});
listdata.Add(new
{
 id = 11,
 name = "China",
 hasChild = true
});
listdata.Add(new
```

```
{
 id = 12,
 pid = 11,
 name = "Guangzhou"
});
listdata.Add(new
{
 id = 13,
 pid = 11,
 name = "Shanghai"
});
listdata.Add(new
{
 id = 14,
 pid = 11,
 name = "Beijing"
});
listdata.Add(new
{
 id = 15,
 pid = 11,
 name = "Shantou"
});
listdata.Add(new
{
 id = 16,
 name = "France",
 hasChild = true
});
listdata.Add(new
{
 id = 17,
 pid = 16,
 name = "Pays de la Loire"
});
listdata.Add(new
{
 id = 18,
 pid = 16,
 name = "Aquitaine"
});
listdata.Add(new
{
 id = 19,
 pid = 16,
 name = "Brittany"
});
listdata.Add(new
{
 id = 20,
 pid = 16,
 name = "Lorraine"
});
listdata.Add(new
{
 id = 21,
 name = "India",
```

```
hasChild = true
});
listdata.Add(new
{
 id = 22,
 pid = 21,
 name = "Assam"
});
listdata.Add(new
{
 id = 23,
 pid = 21,
 name = "Bihar"
});
listdata.Add(new
{
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
});
return View(listdata);
}
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC TreeView control will be rendered in the default web browser.



**Note:** [View Sample in GitHub.](#)

**Note:** You can also explore our [ASP.NET MVC TreeView example](#) to know how to present and manipulate data.

### Data Binding in Treeview Control

The TreeView control provides the option to load data either from local data sources or from remote data services. This can be done through `dataSource` property that is a member of the `fields` property. The `dataSource` property supports array of JavaScript objects and `DataManager`. It also supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of `DataManager` adaptors.

TreeView has `load on demand` (Lazy load), by default. It reduces the bandwidth size when consuming huge data. It loads first level nodes initially, and when parent node is expanded, loads the child nodes based on the `parentID/child` member.

By default, the `loadOnDemand` is set to true. By disabling this property, all the tree nodes are rendered at the beginning itself.

You can use the `dataBound` event to perform actions. This event will be triggered once the data source is populated in the TreeView.

### Local data

To bind local data to the TreeView, you can assign a JavaScript object array to the `dataSource` property. The TreeView control requires three fields (ID, text, and parentID) to render local data source. When mapper fields are not specified, it takes the default values as the mapping fields. Local data source can also be provided as an instance of the `DataManager`. It supports two kinds of local data binding methods.

- Hierarchical data
- Self-referential data

### Hierarchical data

TreeView can be populated with hierarchical data source that contains nested array of JSON objects. You can directly assign hierarchical data to the `dataSource` property, and map all the field members with corresponding keys from the hierarchical data to `fields` property.

In the following example, **code**, **name**, and **countries** columns from hierarchical data have been mapped to **id**, **text**, and **child** fields, respectively.

### CSHTML

```
@Html.EJS().TreeView("listdata").Fields(field=>field.Id("code").Selected("selected").Expanded("expanded").Text("name").Child(ViewBag.Child).DataSource(ViewBag.data)).Render()
```

### HIERARCHICAL.CS

```
public IActionResult LocalData() {
 List<Continents> continents = new List<Continents>();
 List<Countries> countries = new List<Countries>();
 continents.Add(new Continents
 {
 code = "NA",
 name = "North America",
```

```

 expanded=true,
 child = countries,
 });
 countries.Add(new Countries { code = "USA", name = "United
States of America", selected=true});
 countries.Add(new Countries { code = "CUB", name = "Cuba"
});
 countries.Add(new Countries { code = "MEX", name = "Mexico"
});
 List<Countries> countries2 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "AF",
 name = "Africa",
 child = countries2,
 });
 countries2.Add(new Countries { code = "NGA", name = "Nygeria"
});
 countries2.Add(new Countries { code = "EGY", name = "Egypt" });
 countries2.Add(new Countries { code = "ZAF", name = "South
Africa" });
 List<Countries> countries3 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "AS",
 name = "Asia",
 child = countries3,
 });
 countries3.Add(new Countries { code = "CHN", name = "China" });
 countries3.Add(new Countries { code = "IND", name = "India" });
 countries3.Add(new Countries { code = "JPN", name = "Japan" });
 List<Countries> countries4 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "EU",
 name = "Europe",
 child = countries4,
 });
 countries4.Add(new Countries { code = "DNK", name = "Denmark"
});
 countries4.Add(new Countries { code = "FIN", name = "Finland"
});
 countries4.Add(new Countries { code = "AUT", name = "Austria"
});
 List<Countries> countries5 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "SA",
 name = "South America",
 child = countries5,
 });
 countries5.Add(new Countries { code = "BRA", name = "Brazil" });
 countries5.Add(new Countries { code = "COL", name = "Colombia"
});
 countries5.Add(new Countries { code = "ARG", name = "Argentina"
});
 List<Countries> countries6 = new List<Countries>();

```

```

 continents.Add(new Continents
 {
 code = "OC",
 name = "Oceania",
 child = countries6,
 });
 countries6.Add(new Countries { code = "AUS", name = "Australia"
});
 countries6.Add(new Countries { code = "NZL", name = "Newzealand"
});
 countries6.Add(new Countries { code = "WSM", name = "Samoa" });
 List<Countries> countries7 = new List<Countries>();
 continents.Add(new Continents
 {
 code = "AN",
 name = "Antartica",
 child = countries7,
 });
 countries7.Add(new Countries { code = "BVT", name = "Bouvet
Island" });
 countries7.Add(new Countries { code = "ATF", name = "French
Southern Lands" });
 char[] value = { 'c', 'h', 'i', 'l', 'd' };
 string Child = new string(value);
 ViewBag.child = Child;
 ViewBag.data = continents;
 return View();
 }
}

public class Continents
{
 public string code;
 public string name;
 public bool expanded;
 public bool selected;
 public List<Countries> child;
}

public class Countries
{
 public string code;
 public string name;
 public bool expanded;
 public bool selected;
}

```

### Self-referential data

TreeView can be populated from self-referential data structure that contains array of JSON objects with `parentID` mapping.

You can directly assign self-referential data to the `dataSource` property, and map all the field members with corresponding keys from self-referential data to [fields](#) property.



To render the root level nodes, specify the `parentID` as null or no need to specify the `parentID` in `dataSource`.

In the following example, **id**, **pid**, **hasChild**, and **name** columns from self-referential data have been mapped to **id**, **parentID**, **hasChildren**, and **text** fields, respectively.

### CSHTML

```
@model List<object>
@Html.EJS().TreeView("listdata").Fields(field=>
 field.Id("id").ParentID("pid").Selected("selected").
 Expanded("expanded").Text("name").HasChildren("hasChild")
 .DataSource(Model)).Render()
```

### SELFREFERENTIAL.CS

```
public IActionResult LocalData()
{
 List<object> listdata = new List<object>();
 listdata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
 expanded = true
 });
 listdata.Add(new
 {
 id = 2,
 pid = 1,
 name = "New South Wales",
 });
 listdata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Victoria"
 });
 listdata.Add(new
 {
 id = 4,
 pid = 1,
 name = "South Australia"
 });
 listdata.Add(new
 {
 id = 6,
 pid = 1,
 name = "Western Australia",
 });
 listdata.Add(new
 {
 id = 7,
 name = "Brazil",
 hasChild = true
 });
}
```

```
listdata.Add(new
{
 id = 8,
 pid = 7,
 name = "Paraná"
});
listdata.Add(new
{
 id = 9,
 pid = 7,
 name = "Ceará"
});
listdata.Add(new
{
 id = 10,
 pid = 7,
 name = "Acre"
});
listdata.Add(new
{
 id = 11,
 name = "China",
 hasChild = true
});
listdata.Add(new
{
 id = 12,
 pid = 11,
 name = "Guangzhou"
});
listdata.Add(new
{
 id = 13,
 pid = 11,
 name = "Shanghai"
});
listdata.Add(new
{
 id = 14,
 pid = 11,
 name = "Beijing"
});
listdata.Add(new
{
 id = 15,
 pid = 11,
 name = "Shantou"
});
listdata.Add(new
{
 id = 16,
 name = "France",
 hasChild = true
});
listdata.Add(new
{
 id = 17,
```

```
 pid = 16,
 name = "Pays de la Loire"
 });
 listdata.Add(new
 {
 id = 18,
 pid = 16,
 name = "Aquitaine"
 });
 listdata.Add(new
 {
 id = 19,
 pid = 16,
 name = "Brittany"
 });
 listdata.Add(new
 {
 id = 20,
 pid = 16,
 name = "Lorraine"
 });
 listdata.Add(new
 {
 id = 21,
 name = "India",
 hasChild = true
 });
 listdata.Add(new
 {
 id = 22,
 pid = 21,
 name = "Assam"
 });
 listdata.Add(new
 {
 id = 23,
 pid = 21,
 name = "Bihar"
 });
 listdata.Add(new
 {
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
 });
 ViewBag.dataSource = listdata;
 return View();
}
```

### Remote data

TreeView can also be populated from a remote data service with the help of **DataManager** control and **Query** property.

It supports different kinds of data services such as OData, OData V4, Web API, URL, and JSON with the help of **DataManager** adaptors.

You can assign service data as an instance of **DataManager** to the **dataSource** property. To interact with remote data source, you have to provide the endpoint **url**.

The **DataManager** that acts as an interface between the service endpoint and the TreeView requires the following information to interact with service endpoint properly.

- **DataManager->url**: Defines the service endpoint to fetch data.
- **DataManager->adaptor**: Defines the adaptor option. By default, **ODataAdaptor** is used for remote binding.

Adaptor is responsible for processing response and request from/to the service endpoint. The **@syncfusion/ej2-data** package provides some predefined adaptors designed to interact with service endpoints. They are,

- **UrlAdaptor**: Used to interact with remote services. This is the base adaptor for all remote based adaptors.
- **ODataAdaptor**: Used to interact with OData endpoints.
- **ODataV4Adaptor**: Used to interact with OData V4 endpoints.
- **WebApiAdaptor**: Used to interact with Web API created under OData standards.
- **WebMethodAdaptor**: Used to interact with web methods.

In the following example, **ODataV4Adaptor** is used to fetch data from remote services. The **EmployeeID**, **FirstName**, and **EmployeeID** columns from Employees table have been mapped to **id**, **text**, and **hasChildren** fields respectively for first level nodes.

The **OrderID**, **EmployeeID**, and **ShipName** columns from orders table have been mapped to **id**, **parentID**, and **text** fields respectively for second level nodes.

### CSHTML

```
@Html.EJS().TreeView("listdata").Fields(field=>
 field.Query("new
ej.data.Query().from('Employees').select('EmployeeID,FirstName,Title').take(
5)").Id("EmployeeID").ParentID("pid").Selected("selected")
.Text("FirstName").HasChildren("EmployeeID")
.DataSource(dataSource=> {

dataSource.Url("https://services.odata.org/V4/Northwind/Northwind.svc").Adap
tor("ODataV4Adaptor").CrossDomain(true);
}).Child(ViewBag.child)).Render()
```

### REMOTEDATA.CS

```
public IActionResult RemoteData()
{
 TreeViewFieldsSettings childData = new TreeViewFieldsSettings();
 childData.Query = "new
ej.data.Query().from('Orders').select('OrderID,EmployeeID,ShipName').take(5)
";
 childData.Id = "OrderID";
 childData.Text = "ShipName";
 childData.ParentID = "EmployeeID";
```

```

 childData.DataSource = new DataManager
 {
 Url =
 "https://services.odata.org/V4/Northwind/Northwind.svc",
 Adaptor = "ODataV4Adaptor",
 CrossDomain = true
 };
 ViewBag.child = childData;
 return View();
 }

```

## CheckBox

The TreeView control allows you to check more than one node in TreeView without affecting the UI's appearance by enabling the [showCheckBox](#) property. When this property is enabled, checkbox appears before each TreeView node text.

- If one of the child nodes is not in a checked state, then the parent node will be in an intermediate state.
- If all the child nodes are in checked state, then the parent node's state will also be checked.
- If a parent node is checked, then all the child nodes' state will also be checked.

By default, the checkbox state of parent and child nodes are dependent on each other. If you need independent checked state, you can achieve it using the [autoCheck](#) property.

Using the [checkedNodes](#) property, you can set the nodes that need to be checked or get the ID of nodes that are currently checked in the TreeView control.

If you need to prevent the node check action for a particular node, the [nodeChecking](#) event can be used which is triggered before the TreeView node is checked/unchecked. The [nodeChecked](#) event will be triggered when the TreeView node is checked/unchecked successfully.

In the following example, the `showCheckBox` property is enabled.

### CSHTML

```

@Html.EJS().TreeView("treedata").ShowCheckBox(true).Fields(field=>
 field.Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded(
 "expanded")
 .DataSource(ViewBag.dataSource)).Render()

```

### CHECKBOX.CS

```

public IActionResult CheckBox()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new

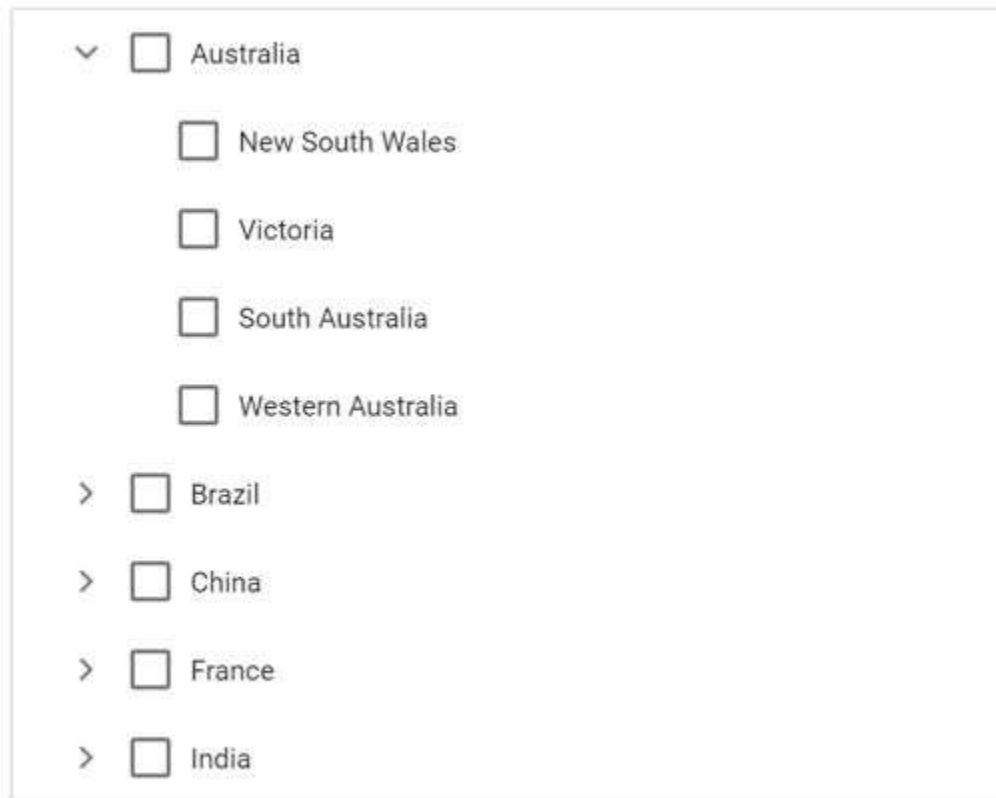
```

```
{
 id = 2,
 pid = 1,
 name = "New South Wales",
});
treedata.Add(new
{
 id = 3,
 pid = 1,
 name = "Victoria"
});
treedata.Add(new
{
 id = 4,
 pid = 1,
 name = "South Australia"
});
treedata.Add(new
{
 id = 6,
 pid = 1,
 name = "Western Australia",
});
treedata.Add(new
{
 id = 7,
 name = "Brazil",
 hasChild = true
});
treedata.Add(new
{
 id = 8,
 pid = 7,
 name = "Paraná"
});
treedata.Add(new
{
 id = 9,
 pid = 7,
 name = "Ceará"
});
treedata.Add(new
{
 id = 10,
 pid = 7,
 name = "Acre"
});
treedata.Add(new
{
 id = 11,
 name = "China",
 hasChild = true
});
treedata.Add(new
{
 id = 12,
 pid = 11,
```

```
 name = "Guangzhou"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 11,
 name = "Shanghai"
 });
 treedata.Add(new
 {
 id = 14,
 pid = 11,
 name = "Beijing"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 11,
 name = "Shantou"
 });
 treedata.Add(new
 {
 id = 16,
 name = "France",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 17,
 pid = 16,
 name = "Pays de la Loire"
 });
 treedata.Add(new
 {
 id = 18,
 pid = 16,
 name = "Aquitaine"
 });
 treedata.Add(new
 {
 id = 19,
 pid = 16,
 name = "Brittany"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 16,
 name = "Lorraine"
 });
 treedata.Add(new
 {
 id = 21,
 name = "India",
 hasChild = true
 });
 treedata.Add(new
```

```
{
 id = 22,
 pid = 21,
 name = "Assam"
});
treedata.Add(new
{
 id = 23,
 pid = 21,
 name = "Bihar"
});
treedata.Add(new
{
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
});
ViewBag.dataSource = treedata;
return View();
}
```

Output be like the below.



#### Checked nodes

You can get or set the checked nodes in TreeView at initial rendering and dynamically by using the [checkedNodes](#) property. It returns the checked nodes' ID as an array.



In the following example, the **New South Wales** and **Western Australia** nodes are checked at initial rendering. If any more nodes are checked, the checked nodes' IDs will be displayed in alert.

### C#HTML

```
@Html.EJS().TreeView("treedata").ShowCheckBox(true).NodeChecked("nodeChecked")
.Fields(field=>

 field.Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded
 ("expanded")

.DataSource(ViewBag.dataSource)).CheckedNodes(ViewBag.checkedNodes).Render()
<script>
 function nodeChecked(args) {
 alert("The checked node's id: " + this.checkedNodes); // To alert
 the checked node's id.
 }
</script>
```

### CHECKBOX.CS

```
public IActionResult CheckBox()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "New South Wales",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Victoria"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "South Australia"
 });
 treedata.Add(new
 {
 id = 6,
 pid = 1,
 name = "Western Australia",
 });
 treedata.Add(new
```

```
{
 id = 7,
 name = "Brazil",
 hasChild = true
});
treedata.Add(new
{
 id = 8,
 pid = 7,
 name = "Paraná"
});
treedata.Add(new
{
 id = 9,
 pid = 7,
 name = "Ceará"
});
treedata.Add(new
{
 id = 10,
 pid = 7,
 name = "Acre"
});
treedata.Add(new
{
 id = 11,
 name = "China",
 hasChild = true
});
treedata.Add(new
{
 id = 12,
 pid = 11,
 name = "Guangzhou"
});
treedata.Add(new
{
 id = 13,
 pid = 11,
 name = "Shanghai"
});
treedata.Add(new
{
 id = 14,
 pid = 11,
 name = "Beijing"
});
treedata.Add(new
{
 id = 15,
 pid = 11,
 name = "Shantou"
});
treedata.Add(new
{
 id = 16,
 name = "France",
```

```
 hasChild = true
 });
 treedata.Add(new
 {
 id = 17,
 pid = 16,
 name = "Pays de la Loire"
 });
 treedata.Add(new
 {
 id = 18,
 pid = 16,
 name = "Aquitaine"
 });
 treedata.Add(new
 {
 id = 19,
 pid = 16,
 name = "Brittany"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 16,
 name = "Lorraine"
 });
 treedata.Add(new
 {
 id = 21,
 name = "India",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 22,
 pid = 21,
 name = "Assam"
 });
 treedata.Add(new
 {
 id = 23,
 pid = 21,
 name = "Bihar"
 });
 treedata.Add(new
 {
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
 });
 ViewBag.dataSource = treedata;
 ViewBag.checkedNodes = new string[] { "2", "6" };
 return View();
}
```

## See Also

- [How to check/uncheck the checkbox on clicking the tree node text](#)

## Node Editing in Treeview Control

The TreeView allows you to edit nodes by setting the [allowEditing](#) property to **true**. To directly edit the nodes in place, **double click** the TreeView node or **select** the node and press **F2** key.

When editing is completed by focus out or by pressing the **Enter** key, the modified node's text saves automatically. If you do not want to save the modified node's text in TreeView node, press **Escape** key. It does not save the edited text to the TreeView node.

- Node editing can also be performed programmatically by using the `beginEdit` method. On passing the node ID or element through this method, the edit textbox will be created for the particular node thus allowing us to edit it.
- If you need to validate or prevent editing, the [nodeEditing](#) event can be used which is triggered before the TreeView node is renamed. On successfully renaming a node the [nodeEdited](#) event will be triggered.

In the following example, the first level node's text cannot be changed, but all other level nodes' text can be changed.

### CSHTML

```
@Html.EJS().TreeView("treedata").AllowEditing(true).Fields(field=>
 field.Selected("is_selected").Id("id").ParentID("pid").Text("name").HasChild
 ren("hasChild").Expanded("expanded")
 .DataSource(ViewBag.dataSource)).Render()
```

### NODEEDITING.CS

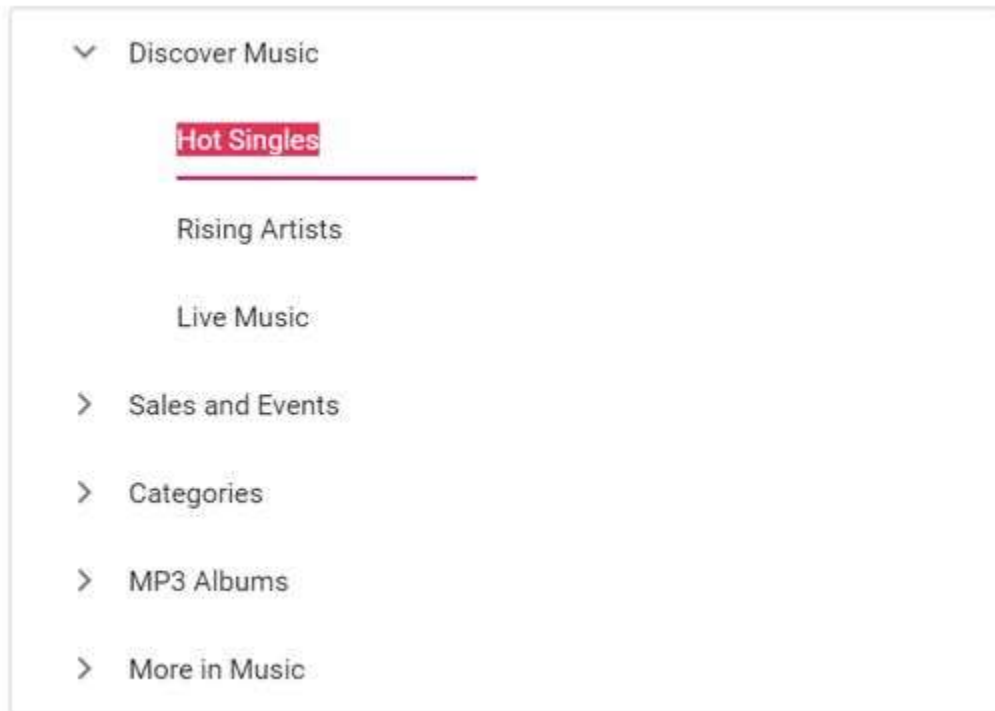
```
public IActionResult NodeEdit()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
}
```

```
});
treedata.Add(new
{
 id = 4,
 pid = 1,
 name = "Live Music"
});
treedata.Add(new
{
 id = 5,
 hasChild = true,
 name = "Sales and Events",
});
treedata.Add(new
{
 id = 6,
 pid=5,
 name = "100 Albums - $5 Each",
});
treedata.Add(new
{
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
});
treedata.Add(new
{
 id = 8,
 pid = 5,
 name = "CD Deals"
});
treedata.Add(new
{
 id = 10,
 hasChild = true,
 name = "Categories"
});
treedata.Add(new
{
 id = 11,
 pid=10,
 name = "Bestselling Albums",
});
treedata.Add(new
{
 id = 12,
 pid = 10,
 name = "New Releases"
});
treedata.Add(new
{
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
});
treedata.Add(new
```

```
{
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
});
treedata.Add(new
{
 id = 15,
 pid = 14,
 name = "Rock"
});
treedata.Add(new
{
 id = 16,
 name = "Gospel",
 pid = 14,
});
treedata.Add(new
{
 id = 17,
 pid = 14,
 name = "Latin Music"
});
treedata.Add(new
{
 id = 18,
 pid = 14,
 name = "Jazz"
});
treedata.Add(new
{
 id = 19,
 hasChild = true,
 name = "More in Music"
});
treedata.Add(new
{
 id = 20,
 pid = 19,
 name = "Music Trade-In"
});
treedata.Add(new
{
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
});
treedata.Add(new
{
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
});

ViewBag.dataSource = treedata;
return View();
}
```

Output be like the below.



See Also

- [How to validate the text when renaming the tree node](#)
- [How to process the tree node operations using context menu](#)

### Multi Selection in TreeView Control

Selection provides an interactive support and highlights the node that you select. Selection can be done through simple mouse down or keyboard interaction.

The TreeView also supports selection of multiple nodes by setting [allowMultiSelection](#) to **true**.

To multi-select, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes. In the following example, the `allowMultiSelection` property is enabled.

**Note:** Multi selection is not applicable through touch interactions.

#### CSHTML

```
@Html.EJS().TreeView("treedata").AllowMultiSelection(true).Fields(field=>
 field.Selected("is_selected").Id("id").ParentID("pid").Text("name").HasChild
 ren("hasChild").Expanded("expanded")
 .DataSource(ViewBag.dataSource)).Render()
```

#### MULTISELECTION.CS

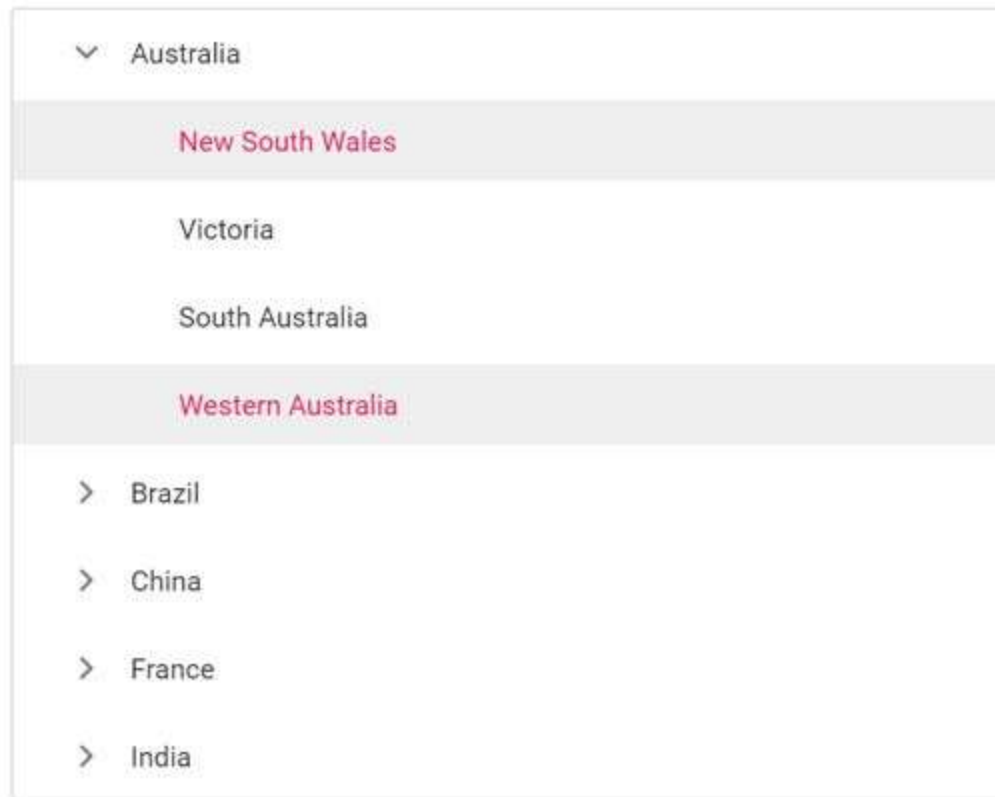
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TreeView
{
 public partial class TreeViewController : Controller
 {
 public IActionResult MultiSelection()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "New South Wales",
 is_selected=true
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Victoria"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "South Australia"
 });
 treedata.Add(new
 {
 id = 6,
 pid = 1,
 name = "Western Australia",
 is_selected = true
 });
 treedata.Add(new
 {
 id = 7,
 name = "Brazil",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 8,
 pid = 7,
 name = "Paraná"
 });
 }
 }
}
```



```
});
treedata.Add(new
{
 id = 9,
 pid = 7,
 name = "Ceará"
});
treedata.Add(new
{
 id = 10,
 pid = 7,
 name = "Acre"
});
treedata.Add(new
{
 id = 11,
 name = "China",
 hasChild = true
});
treedata.Add(new
{
 id = 12,
 pid = 11,
 name = "Guangzhou",
});
treedata.Add(new
{
 id = 13,
 pid = 11,
 name = "Shanghai"
});
treedata.Add(new
{
 id = 14,
 pid = 11,
 name = "Beijing"
});
treedata.Add(new
{
 id = 15,
 pid = 11,
 name = "Shantou"
});
treedata.Add(new
{
 id = 16,
 name = "France",
 hasChild = true
});
treedata.Add(new
{
 id = 17,
 pid = 16,
 name = "Pays de la Loire"
});
treedata.Add(new
{
```

```
 id = 18,
 pid = 16,
 name = "Aquitaine"
 });
 treedata.Add(new
 {
 id = 19,
 pid = 16,
 name = "Brittany"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 16,
 name = "Lorraine"
 });
 treedata.Add(new
 {
 id = 21,
 name = "India",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 22,
 pid = 21,
 name = "Assam"
 });
 treedata.Add(new
 {
 id = 23,
 pid = 21,
 name = "Bihar"
 });
 treedata.Add(new
 {
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
 });
 ViewBag.dataSource = treedata;
 return View();
}
}
```

Output be like the below.



### Selected nodes

You can get or set the selected nodes in TreeView at initial rendering and dynamically by using the [selectedNodes](#) property. It will return the selected node's ID as an array.

- The [nodeselecting](#) event is triggered before a node is selected/unselected which can be used to prevent the selection.
- The [nodeSelected](#) event is triggered once a node is successfully selected/unselected.

In the following example, **New South Wales** and **Western Australia** nodes are selected at initial rendering. When a node is selected, the selected node's ID is displayed in alert.

### CSHTML

```
@Html.EJS().TreeView("treedata").AllowMultiSelection(true).NodeSelected("nodeSelected").Fields(field=>

 field.Selected("is_selected").Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded("expanded")

.DataSource(ViewBag.dataSource)).SelectedNodes(ViewBag.selectedNodes).Render()

<script>
function nodeselect() {
 alert("The selected node's id: " + this.selectedNodes);
}
</script>
```

**SELECTIONNODE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TreeView
{
 public partial class TreeViewController : Controller
 {
 public IActionResult MultiSelection()
 {
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "New South Wales"
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Victoria"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "South Australia"
 });
 treedata.Add(new
 {
 id = 6,
 pid = 1,
 name = "Western Australia",
 is_selected = true
 });
 treedata.Add(new
 {
 id = 7,
 name = "Brazil",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 8,
```

```
 pid = 7,
 name = "Paraná"
 });
 treedata.Add(new
 {
 id = 9,
 pid = 7,
 name = "Ceará"
 });
 treedata.Add(new
 {
 id = 10,
 pid = 7,
 name = "Acre"
 });
 treedata.Add(new
 {
 id = 11,
 name = "China",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 12,
 pid = 11,
 name = "Guangzhou",
 });
 treedata.Add(new
 {
 id = 13,
 pid = 11,
 name = "Shanghai"
 });
 treedata.Add(new
 {
 id = 14,
 pid = 11,
 name = "Beijing"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 11,
 name = "Shantou"
 });
 treedata.Add(new
 {
 id = 16,
 name = "France",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 17,
 pid = 16,
 name = "Pays de la Loire"
 });
});
```

```

 treedata.Add(new
 {
 id = 18,
 pid = 16,
 name = "Aquitaine"
 });
 treedata.Add(new
 {
 id = 19,
 pid = 16,
 name = "Brittany"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 16,
 name = "Lorraine"
 });
 treedata.Add(new
 {
 id = 21,
 name = "India",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 22,
 pid = 21,
 name = "Assam"
 });
 treedata.Add(new
 {
 id = 23,
 pid = 21,
 name = "Bihar"
 });
 treedata.Add(new
 {
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
 });
 ViewBag.dataSource = treedata;
 ViewBag.selectedNodes = new string[] { "2", "3" };
 return View();
 }
}

```

## Drag and Drop

The TreeView control allows you to drag and drop any node by setting [allowDragAndDrop](#) to **true**. Nodes can be dragged and dropped at all levels of the same TreeView.

The dragged nodes can be dropped at any level by indicator lines with **line**, **plus/minus**, and **restrict** icons. It represents the exact position where the node is to be dropped as sibling or child.

The following table explains the usage of indicator icons.

| Icons                  | Description                                                                   |
|------------------------|-------------------------------------------------------------------------------|
| ----- -----            |                                                                               |
| Plus icon              | Indicates that the dragged node is to be added as child of target node.       |
| Minus or restrict icon | Indicates that the dragged node is not to be dropped at the hovered region.   |
| In between icon        | Indicates that the dragged node is to be added as siblings of hovered region. |

- If you need to prevent dragging action for a particular node, the [nodeDragStart](#) event can be used which is triggered when the node drag is started. If you need to prevent dropping action for a particular node, the [nodeDragStop](#) event can be used which is triggered when the drag is stopped.
- The [nodeDragging](#) event is triggered when the TreeView node is being dragged. You can customize the cloned element in this event.
- The [nodeDropped](#) event is triggered when the TreeView node is dropped on the target element successfully.

In the following sample, the [allowDragAndDrop](#) property is enabled.

#### CSHTML

```
@Html.EJS().TreeView("treedata").AllowDragAndDrop(true).Fields(field=>
 field.Selected("is_selected").Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded("expanded")
 .DataSource(ViewBag.dataSource).Render()
```

#### DRAGANDDROP.CS

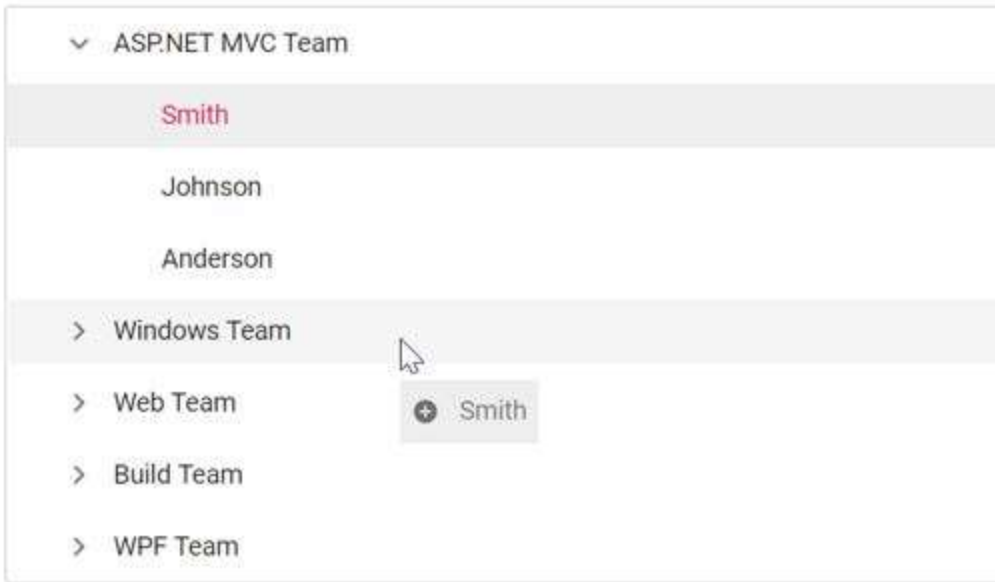
```
public IActionResult DragDrop()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "ASP.NET MVC Team",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Smith",
 is_selected=true
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Johnson",
 is_selected = true
 });
}
```

```
});
treedata.Add(new
{
 id = 4,
 pid = 1,
 name = "Anderson"
});
treedata.Add(new
{
 id = 6,
 hasChild = true,
 name = "Windows Team",
});
treedata.Add(new
{
 id = 7,
 pid=6,
 name="Clark"
});
treedata.Add(new
{
 id = 8,
 pid = 6,
 name = "Wright"
});
treedata.Add(new
{
 id = 9,
 pid = 6,
 name = "Lopez"
});
treedata.Add(new
{
 id = 10,
 hasChild = true,
 name = "Web Team"
});
treedata.Add(new
{
 id = 11,
 pid=10,
 name = "Joshua",
});
treedata.Add(new
{
 id = 12,
 pid = 10,
 name = "Matthew"
});
treedata.Add(new
{
 id = 13,
 pid = 10,
 name = "David"
});
```



```
treedata.Add(new
{
 id = 14,
 hasChild = true,
 name = "Build Team"
});
treedata.Add(new
{
 id = 15,
 pid = 14,
 name = "Ryan"
});
treedata.Add(new
{
 id = 16,
 pid=14,
 name = "Justin",
});
treedata.Add(new
{
 id = 17,
 pid = 14,
 name = "Robert"
});
treedata.Add(new
{
 id = 18,
 hasChild=true,
 name = "WPF Team"
});
treedata.Add(new
{
 id = 19,
 pid = 18,
 name = "Brown"
});
treedata.Add(new
{
 id = 20,
 pid = 18,
 name = "Johnson"
});
treedata.Add(new
{
 id = 21,
 pid = 18,
 name = "Miller"
});
ViewBag.dataSource = treedata;
return View();
}
```

Output be like the below.



### Multiple-node drag and drop

To drag and drop more than one node, you should enable the [allowMultiSelection](#) property along with the [allowDragAndDrop](#) property.

To perform multi-selection, press and hold **CTRL** key and click the desired nodes. To select range of nodes, press and hold the **SHIFT** key and click the nodes.

In the following sample, the [allowMultiSelection](#) property is enabled along with the [allowDragAndDrop](#) property.

### CSHTML

```
@Html.EJS().TreeView("treedata").AllowDragAndDrop(true).AllowMultiSelection(
true).Fields(field=>

 field.Selected("is_selected").Id("id").ParentID("pid").Text("name").HasChild
ren("hasChild").Expanded("expanded")
 .DataSource(ViewBag.dataSource).Render()
```

### DRAGANDDROP.CS

```
public IActionResult DragDrop()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "ASP.NET MVC Team",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
```

```
 name = "Smith",
 is_selected=true
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Johnson",
 is_selected = true
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Anderson"
 });
 treedata.Add(new
 {
 id = 6,
 hasChild = true,
 name = "Windows Team",
 });
 treedata.Add(new
 {
 id = 7,
 pid=6,
 name="Clark"
 });
 treedata.Add(new
 {
 id = 8,
 pid = 6,
 name = "Wright"
 });
 treedata.Add(new
 {
 id = 9,
 pid = 6,
 name = "Lopez"
 });
 treedata.Add(new
 {
 id = 10,
 hasChild = true,
 name = "Web Team"
 });
 treedata.Add(new
 {
 id = 11,
 pid=10,
 name = "Joshua",
 });
 treedata.Add(new
 {
 id = 12,
```

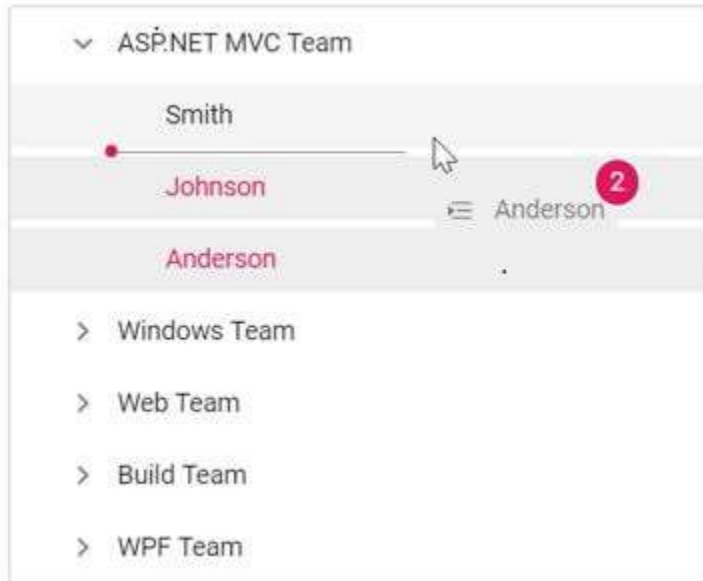
```
 pid = 10,
 name = "Matthew"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 10,
 name = "David"
 });
 treedata.Add(new
 {
 id = 14,
 hasChild = true,
 name = "Build Team"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 14,
 name = "Ryan"
 });
 treedata.Add(new
 {
 id = 16,
 pid=14,
 name = "Justin",
 });
 treedata.Add(new
 {
 id = 17,
 pid = 14,
 name = "Robert"
 });
 treedata.Add(new
 {
 id = 18,
 hasChild=true,
 name = "WPF Team"
 });
 treedata.Add(new
 {
 id = 19,
 pid = 18,
 name = "Brown"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 18,
 name = "Johnson"
 });
 treedata.Add(new
 {
 id = 21,
 pid = 18,
 name = "Miller"
```

```

 });
 ViewBag.dataSource = treedata;
 return View();
}

```

Output be like the below.



See Also

- [How to restrict the drag-and-drop for particular tree nodes](#)

## Template

The TreeView control allows you to customize the look of TreeView nodes by using the [nodeTemplate](#) property. This property accepts either `template string` or HTML element ID.

In the following sample, employee information such as employee photo, name, and designation have been included using the `nodeTemplate` property.

The template expression should be provided inside the `${...}` interpolation syntax.

### CSHTML

```

<div class="treecontainer">
 @Html.EJS().TreeView("treedata").CssClass("custom").NodeTemplate("<div><img
class=\"eimage\"
src=\"https://ej2.syncfusion.com/demos/src/images/employees/${Image}.png\"
alt=\"employee\"/><div class=\"ename\"> ${name} </div><div class=\"ejob\">
${Job} </div></div>").Fields(field=>

 field.Selected("is_selected").Id("id").ParentID("pid").Text("name").HasChild
ren("HasChild").Expanded("Expanded")
 .DataSource(ViewBag.data).Render()
</div>
<style>
 .custom.e-treeview .e-fullrow {

```

```

 height: 72px;
 }
 .custom.e-treeview .e-list-text {
 line-height: normal;
 }
 .eimage {
 float: left;
 padding: 11px 16px 11px 0;
 height: 48px;
 width: 48px;
 box-sizing: content-box;
 }
 .ename {
 font-size: 16px;
 padding: 14px 0 0;
 }
 .ejob {
 font-size: 14px;
 opacity: .87;
 }
}
</style>

```

## TEMPLATEMODEL.CS

```

public class TreeviewTemplate
{
 public string name { get; set; }
 public string count { get; set; }
 public int id { get; set; }
 public int pid { get; set; }
 public bool HasChild { get; set; }
 public bool Expanded { get; set; }
 public bool Selected { get; set; }
 public string Image { get; set; }
 public string Job { get; set; }
 public List<TreeviewTemplate> getTreeviewTemplate()
 {
 List<TreeviewTemplate> localData = new
List<TreeviewTemplate>();
 localData.Add(new TreeviewTemplate { id = 1, name = "Steven
Buchanan", HasChild = true, Expanded=true, Image = "10", Job= "CEO" });
 localData.Add(new TreeviewTemplate { id = 2, pid = 1, name =
"Laura Callahan", count = "4", Image = "2", Job= "Product Manager", HasChild
= true });
 localData.Add(new TreeviewTemplate { id = 3, pid = 2, name =
"Andrew Fuller", Image = "7", Job= "Team Lead", HasChild= true });
 localData.Add(new TreeviewTemplate { id = 4, pid = 3, name =
"Anne Dodsworth", count = "6", Image = "1", Job= "Developer" });
 localData.Add(new TreeviewTemplate { id = 5, pid= 1, name =
"Nancy Davolio", HasChild = true, Image = "4", Job= "Product Manager" });
 localData.Add(new TreeviewTemplate { id = 6, pid = 5, name =
"Michael Suyama", count = "20", Image = "9", Job= "Team Lead", HasChild =
true });
 localData.Add(new TreeviewTemplate { id = 7, pid = 6, name =
"Robert King", count = "5", Image = "8", Job= "Developer" });
 }
}

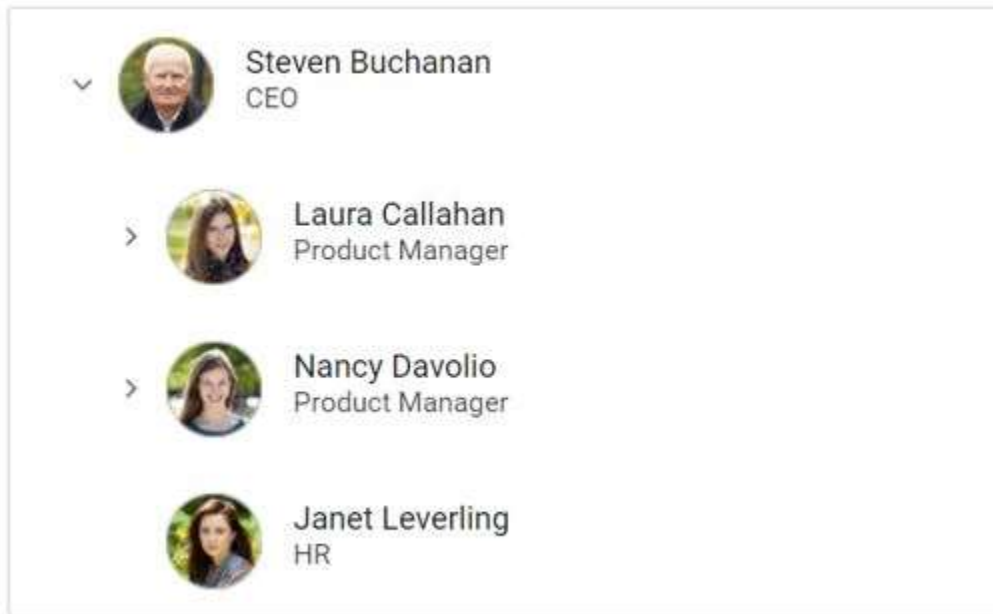
```

```

 localData.Add(new TreeviewTemplate { id = 8, pid = 7, name =
"Margaret Peacock", Image = "6" , Job= "Developer" });
 localData.Add(new TreeviewTemplate { id = 9, pid = 1, name =
"Janet Leverling", Image = "3", Job= "HR" });
 return localData;
 }
}

```

Output be like the below.



See Also

- [How to customize the expand and collapse icons](#)
- [How to customize the tree nodes based on levels](#)

### Accessibility in ASP.NET MVC TreeView component

The TreeView component followed the accessibility guidelines and standards, including [ADA](#), [Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the TreeView component is outlined below.

| Accessibility Criteria | Compatibility |

| -- | -- |

| [WCAG 2.2](#) Support |  |

| [Section 508](#) Support |  |

| Screen Reader Support |  |

```

| Right-To-Left Support | |

| Color Contrast | |

| Mobile Device Support | |

| Keyboard Navigation Support | |

| Accessibility Checker Validation | |

| Axe-core Accessibility Validation | |

<style>

.post .post-content img {
display: inline-block;
margin: 0.5em 0;
}

</style>

<div> - All
features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

```

#### WAI-ARIA attributes

The TreeView component followed the [WAI-ARIA](#) patterns to meet the accessibility. The following ARIA attributes are used in the TreeView component:

| Attributes                        | Purpose                                                                                            |
|-----------------------------------|----------------------------------------------------------------------------------------------------|
| ---                               | ---                                                                                                |
| <code>role=tree</code>            | All tree nodes are contained within the element.                                                   |
| <code>role=treeitem</code>        | Specifies the role of each tree node in a selectable TreeView and its containment within the tree. |
| <code>role=group</code>           | Specifies the role of each parent node container.                                                  |
| <code>role=checkbox</code>        | Indicates checkbox control along with treeitem element.                                            |
| <code>aria-multiselectable</code> | Indicates whether the TreeView enables multiple selection or not.                                  |
| <code>aria-expanded</code>        | Indicates whether the parent node has expanded or not.                                             |



|                              |                                                                        |
|------------------------------|------------------------------------------------------------------------|
| <b>aria-selected</b>         | Indicates the selected node.                                           |
| <b>aria-grabbed</b>          | Indicates the selected state on drag-and-drop of node.                 |
| <b>aria-level</b>            | Indicates the level of node in TreeView.                               |
| <b>aria-checked</b>          | Indicates the current checked state of TreeView checkbox.              |
| <b>aria-label</b>            | Indicates the contextual message for the TreeView checkbox.            |
| <b>aria-activedescendant</b> | Identifies the currently active element when focusing on the TreeView. |
| <b>aria-disabled</b>         | Indicates element is perceivable but disabled.                         |

### Keyboard interaction

The TreeView component followed the [keyboard interaction](#) guideline, making it easy for people who use assistive technologies (AT) and those who completely rely on keyboard navigation. The following keyboard shortcuts are supported by the TreeView component.

| Interaction Keys | Description                                                |
|------------------|------------------------------------------------------------|
| ----- -----      |                                                            |
| Arrow Up         | Goes to the previous node.                                 |
| Arrow Down       | Goes to the next node.                                     |
| Arrow Right      | Expands the current node.                                  |
| Arrow Left       | Collapses the current node.                                |
| Home             | Goes to the first node.                                    |
| End              | Goes to the last node.                                     |
| F2               | Edits the focused node.                                    |
| Esc              | Focuses out the edit state without saving the edited text. |
| Enter            | Selects the focused node/saves the edited text.            |
| Space            | Checks the current node.                                   |
| Ctrl + A         | Selects all nodes.                                         |

### Ensuring accessibility

The TreeView component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the TreeView component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the TreeView component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

## How To

### Customize the expand and collapse icons

You can customize TreeView expand and collapse icons by using the [cssClass](#) property of TreeView.

Refer to the sample to customize expand/collapse icons.

### CSHTML

```
@Html.EJS().TreeView("treedata").CssClass("custom").Fields(field=>
 field.Id("nodeId").ParentID("pid").Text("nodeText").HasChildren("hasChild").
 Expanded("expanded")
 .Child(ViewBag.child).DataSource(ViewBag.dataSource)).Render()

<style>
 .custom .e-list-item .e-icons {
 font-family: "Customize-icon";
 }
 .custom.e-treeview .e-list-item .e-icon-expandable::before, .custom.e-
 treeview .e-list-item .e-icon-collapsible:before {
 content: '\e700';
 font-size: 12px;
 }
 @@font-face {
 font-family: 'Customize-icon';
 src: url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMj0gSRcAAAEoAAAAVmNtYXNlE0daAAABjAAAADhnbHlmcYq
IngAAAcwAAAD8aGVhZBWT124AAADQAAAAANmhoZWEHmAntAAAArAAAACRobXR4C9AAAAAAYAAAA
MbG9jYQBAAH4AAAHEAAAACG1heHABEAAxAAABCAAAACBuYW1l/qscPAAAASgAAAJ5cG9zdIPGFvo
AAAVEAAAAVgABAAADUv9qAFoEAAAA//8D6QABAAAAAAAAAAAAAAAAAAwABAAAAQAAlKcGUl8
PPPUACwPoAAAAANlGSVAAAAAA2UZJUAAAAAAD6QPpAAAACAACAAAAAAAAAAAAADACUAAwAAAAA
AAgAAAAoACgAAAP8AAAAAAAAAAQpWAZAABQAAAnoCvAAAAIwCegK8AAAB4AAxAQIAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnAQNS/2oAWgPpAJYAAAABAAAAAAAAABAAAAAPoAAA
D6AAAAAAAAAgAAAAAAAUAAAAQAABQABAAkAAAAABAAEAAEAAOCB//8AAOCa//8AAAAABAAQAAAA
BAAIAAAAAEAafgADAAAAAPpA+kACAACWACQAAAEhFSEHmZcnIyUWEAcGICcmEDc+ATIWBQYQFxy
gNzYQJy4BIgYCMf6kAWqUqMK8rgF+goKK/qCEfn5Coquf/amRkZoBkpgRkUq3xLcCKmSTybt4if6
ghYKChQFgiUJBQRma/m6akZGaAZKaSElJAAMAAAAA+gD6QAGABQAIgAAASMXNyMRIyUWEAcGICc
mEDc+ATIWBQYQFxygNzYQJy4BIgYBvRlp6JmGAW6BgYf+oYiBgUGhqqH9qZOTmgGOMPOTsrBctgG
y6ekBbwuI/qGHgYGIaV6IQEFBFpr+cZmTk5oBj5lKSUKAAAAAABIA3gABAAAAAAAAAAAAEAAAABAA
AAAABAA4AAQABAAAAAAACAAcADwABAAAAAADAA4AFgABAAAAAAEAA4AJAABAAAAAAFAAsAMgA
BAAAAAAGAA4APQABAAAAAAAKACwASwABAAAAAALABIAdwADAAEEECQAAAAIAiQADAAEEECQABABw
AiwADAAEEECQACAA4APwADAAEEECQADABwAtQADAAEEECQAEABwA0QADAAEEECQAFABYA7QADAAEEECQA
GABwBAwADAAEEECQAKAFgBHwADAAEEECQALACQBdyBDdXN0b21pemUtaWNvblJlZ3VsYXJldXN0b21
pemUtaWNvbWl6ZS1pY29uVmVyc2lubiAxLjBDdXN0b21pemUtaWNvbWl6ZS1pY29uVmVyc2lubiAxLj
hdGVkIHVzaW5nIFN5bmNmdXNpb24gTWV0cm8gU3RlZGlvZ3d3LnN5bmNmdXNpb24uY29tACAAQwB
1AHMADABvAG0AaQB6AGUALQBpAGMABwBuAFIAZQBnAHUAbABhAHIAQwB1AHMADABvAG0AaQB6AGU
ALQBpAGMABwBuAEMADQBzAHQABwBtAGkAegBlAC0AaQBjAG8AbgBWAGUAcgBzAGkABwBuACAAMQA
uADAAQwB1AHMADABvAG0AaQB6AGUALQBpAGMABwBuAEYABwBuAHQAIAABnAGUAbgBlAHIAIYQB0AGU
AZAAgAHUAcwBpAG4AZwAgAFMAeQBwAGMAZgB1AHMAaQBvAG4AIABNAGUAdABYAG8AIABTAHQAdQB
kAGkABwB3AHcAdwAuAHMAeQBwAGMAZgB1AHMAaQBvAG4ALgBjAG8AbQAAAAACAAAAAAAAAAoAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAMBAGEDAQQAFylhcnJvdyljaXJjbGUtcmlnaHQQtXzAxZlhcJ
vdyljaXJjbGUtZG93bgAAAAA=) format('trueType');
 font-weight: normal;
 font-style: normal;
 }
```

</style>

**CUSTOMIZE.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TreeView
{
 public partial class TreeViewController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 List<Parentitem> parentitem = new List<Parentitem>();
 List<Childitem> childitem1 = new List<Childitem>();
 List<SubChilditem> subchilditem1 = new List<SubChilditem>();
 parentitem.Add(new Parentitem
 {
 nodeId = "01",
 nodeText = "Local Disk (C:)",
 expanded=true,
 child = childitem1,
 });
 childitem1.Add(new Childitem { nodeId = "01-01", nodeText =
"Program Files", child=subchilditem1 });
 subchilditem1.Add(new SubChilditem { nodeId = "01-01-01",
nodeText = "Windows NT" });
 subchilditem1.Add(new SubChilditem { nodeId = "01-01-02",
nodeText = "Windows Mail" });
 subchilditem1.Add(new SubChilditem { nodeId = "01-01-03",
nodeText = "Windows Photo Viewer" });
 List<SubChilditem> subchilditem2 = new List<SubChilditem>();
 childitem1.Add(new Childitem { nodeId = "01-02", nodeText =
"Users", expanded=true, child = subchilditem2 });
 subchilditem2.Add(new SubChilditem { nodeId = "01-02-01",
nodeText = "Smith" });
 subchilditem2.Add(new SubChilditem { nodeId = "01-02-02",
nodeText = "Public" });
 subchilditem2.Add(new SubChilditem { nodeId = "01-02-03",
nodeText = "Admin" });
 List<SubChilditem> subchilditem3 = new List<SubChilditem>();
 childitem1.Add(new Childitem { nodeId = "01-03", nodeText =
"Windows", child = subchilditem3 });
 subchilditem3.Add(new SubChilditem { nodeId = "01-03-01",
nodeText = "Boot" });
 subchilditem3.Add(new SubChilditem { nodeId = "01-03-02",
nodeText = "FileManager" });
 subchilditem3.Add(new SubChilditem { nodeId = "01-03-03",
nodeText = "System32" });
 List<Childitem> childitem2 = new List<Childitem>();
 parentitem.Add(new Parentitem
 {
 nodeId = "02",
 nodeText = "Local Disk (D:)",
 });
 }
 }
}

```

```

 child = childitem2,
 });
 List<SubChilditem> subchilditem4 = new List<SubChilditem>();
 childitem2.Add(new Childitem { nodeId = "02-01", nodeText =
"Personals", child=subchilditem4});
 subchilditem4.Add(new SubChilditem { nodeId = "02-01-01",
nodeText = "My photo.png" });
 subchilditem4.Add(new SubChilditem { nodeId = "02-01-02",
nodeText = "Rental document.docx" });
 subchilditem4.Add(new SubChilditem { nodeId = "02-01-03",
nodeText = "Pay slip.pdf" });
 List<SubChilditem> subchilditem5 = new List<SubChilditem>();
 childitem2.Add(new Childitem { nodeId = "02-02", nodeText =
"Projects", child=subchilditem5 });
 subchilditem5.Add(new SubChilditem { nodeId = "02-02-01",
nodeText = "ASP Application " });
 subchilditem5.Add(new SubChilditem { nodeId = "02-02-02",
nodeText = "TypeScript Application" });
 subchilditem5.Add(new SubChilditem { nodeId = "02-02-03",
nodeText = "React Application" });

 List<SubChilditem> subchilditem6 = new List<SubChilditem>();
 childitem2.Add(new Childitem { nodeId = "02-02", nodeText =
"Office", child = subchilditem6 });
 subchilditem6.Add(new SubChilditem { nodeId = "02-03-01",
nodeText = "Work details.docx " });
 subchilditem6.Add(new SubChilditem { nodeId = "02-03-02",
nodeText = "Weekly report.docx" });
 subchilditem6.Add(new SubChilditem { nodeId = "02-03-03",
nodeText = "Wish list.csv" });
 List<Childitem> childitem3 = new List<Childitem>();
 parentitem.Add(new Parentitem
 {
 nodeId = "03",
 nodeText = "Local Disk (E:)",
 child = childitem3,
 });

 List<SubChilditem> subchilditem7 = new List<SubChilditem>();
 childitem3.Add(new Childitem { nodeId = "03-01", nodeText =
"Pictures", child=subchilditem7});
 subchilditem7.Add(new SubChilditem { nodeId = "03-01-01",
nodeText = "Wind.jpg " });
 subchilditem7.Add(new SubChilditem { nodeId = "03-01-02",
nodeText = "Stone.jpg" });
 subchilditem7.Add(new SubChilditem { nodeId = "03-01-03",
nodeText = "Home.jpg" });

 List<SubChilditem> subchilditem8 = new List<SubChilditem>();
 childitem3.Add(new Childitem { nodeId = "03-02", nodeText =
"Documents", icon = "docx", child=subchilditem8});
 subchilditem8.Add(new SubChilditem { nodeId = "03-02-01",
nodeText = "Environment Pollution.docx " });
 subchilditem8.Add(new SubChilditem { nodeId = "03-02-02",
nodeText = "Global Warming.ppt" });
 subchilditem8.Add(new SubChilditem { nodeId = "03-02-03",
nodeText = "Social Network.pdf" });

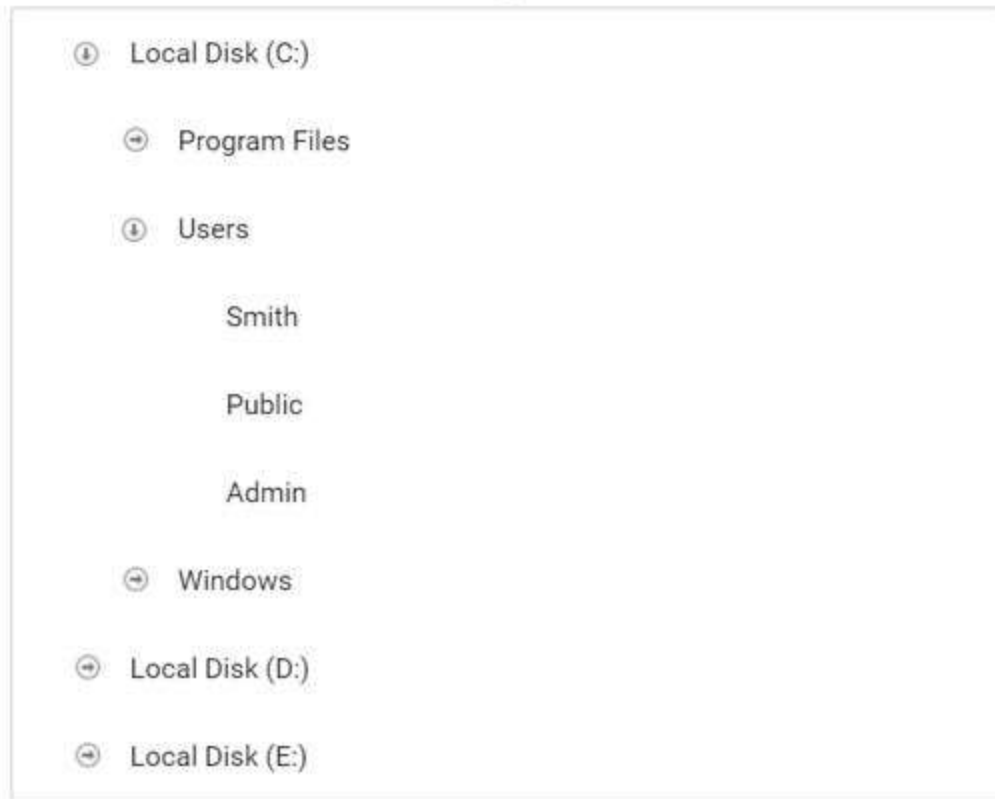
```

```

 List<SubChilditem> subchilditem9 = new List<SubChilditem>();
 childitem3.Add(new Childitem { nodeId = "03-03", nodeText =
"Study Materials",child=subchilditem9 });
 subchilditem9.Add(new SubChilditem { nodeId = "03-03-01",
nodeText = "UI-Guide.pdf" });
 subchilditem9.Add(new SubChilditem { nodeId = "03-03-02",
nodeText = "Tutorials.zip" });
 subchilditem9.Add(new SubChilditem { nodeId = "03-03-03",
nodeText = "TypeScript.7z" });
 ViewBag.dataSource = parentitem;
 char[] value = { 'c', 'h', 'i', 'l', 'd' };
 string Child = new string(value);
 ViewBag.child = Child;
 return View();
 }
}
}
public class Parentitem
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<Childitem> child;
}
public class Childitem
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<SubChilditem> child;
}
public class SubChilditem
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
}

```

Output be like the below.



### Process the tree node operations using context menu

You can integrate the context menu with 'TreeView' control in order to perform the tree view related operations like add, remove and renaming node.

Following is an example which demonstrates the above cases which are used to manipulate tree view operations in the 'select' event of context menu.

### CSHTML

```
@Html.EJS().TreeView("treedata").CssClass("custom").Fields(field=>
 field.Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded(
 "expanded")

 .Selected("is_selected").HtmlAttributes("hasAttribute").DataSource(ViewBag.d
 ataSource)).NodeClicked("nodeClicked").Render()
 @Html.EJS().ContextMenu("menu").Target("#treedata").Select("menuClick").Befo
 reOpen("beforeOpen").Items(ViewBag.menuItems).Render()
<script>
 var index = 1;
 function nodeClicked(args) {
 if (args.event.which === 3) {
 var treeObj =
 document.getElementById('treedata').ej2_instances[0];
 treeObj.selectedNodes = [args.node.getAttribute('data-uid')];
 }
 }
 function menuClick(args) {
 var treeObj = document.getElementById('treedata').ej2_instances[0];
```

```

var menuObj = document.getElementById('menu').ej2_instances[0];
var targetNodeId = treeObj.selectedNodes[0];
if (args.item.text == "Add New Item") {
 var nodeId = "tree_" + index;
 var item = { id: nodeId, name: "New Folder" };
 treeObj.addNodes([item], targetNodeId, null);
 index++;
 treeObj.fields.dataSource.push(item);
 treeObj.beginEdit(nodeId);
}
else if (args.item.text == "Remove Item") {
 treeObj.removeNodes([targetNodeId]);
}
else if (args.item.text == "Rename Item") {
 treeObj.beginEdit(targetNodeId);
}
}
function beforeOpen(args) {
 var treeObj = document.getElementById('treedata').ej2_instances[0];
 var menuObj = document.getElementById('menu').ej2_instances[0];
 var targetNodeId = treeObj.selectedNodes[0];
 var targetNode =
document.getElementById('treedata').querySelector('[data-uid="' +
targetNodeId + '"]');
 if (targetNode.classList.contains('remove')) {
 menuObj.enableItems(['Remove Item'], false);
 }
 else {
 menuObj.enableItems(['Remove Item'], true);
 }
 if (targetNode.classList.contains('rename')) {
 menuObj.enableItems(['Rename Item'], false);
 }
 else {
 menuObj.enableItems(['Rename Item'], true);
 }
}
</script>
}
<style>
 .control_wrapper {
 max-width: 500px;
 margin: 0 auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
 }
</style>

```

## CONTEXTMENU.CS

```

using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TreeView
{
 public partial class TreeViewController : Controller
 {

```

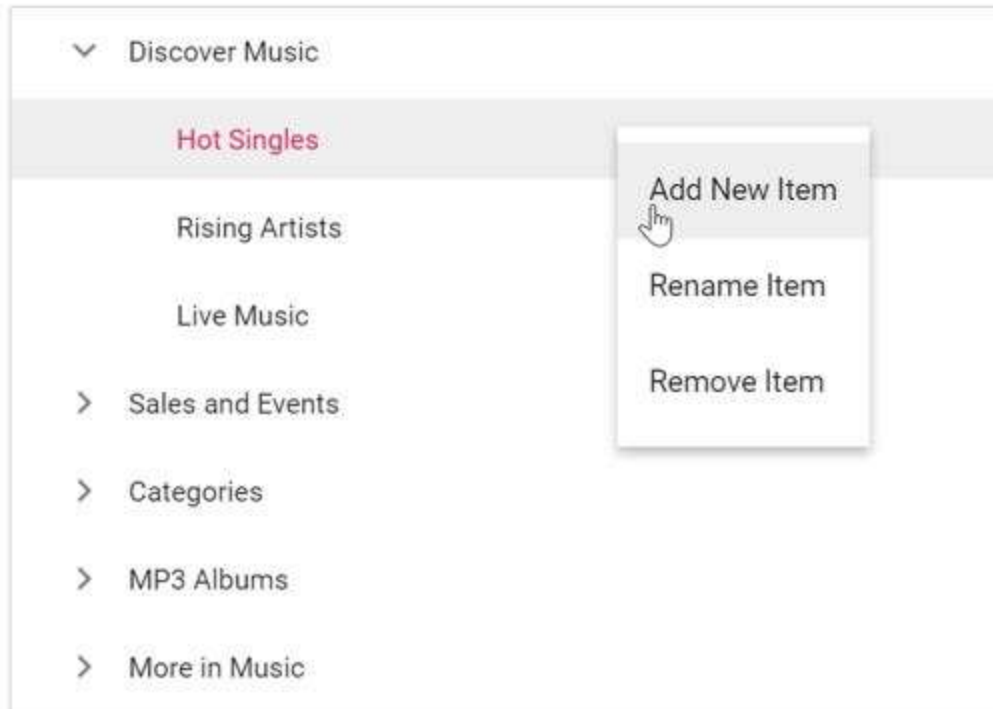
```
Dictionary<string, string> htmlAttribute = new Dictionary<string,
string>();
public IActionResult ContextMenu()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true,
 htmlAttribute = new Dictionary<string, string>() {{ "class"
, "remove rename" }}
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Live Music"
 });
 treedata.Add(new
 {
 id = 5,
 hasChild = true,
 name = "Sales and Events",
 });
 treedata.Add(new
 {
 id = 6,
 pid=5,
 name = "100 Albums - $5 Each",
 });
 treedata.Add(new
 {
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
 });
 treedata.Add(new
 {
 id = 8,
 pid = 5,
 name = "CD Deals"
 });
 treedata.Add(new
```



```
{
 id = 10,
 hasChild = true,
 name = "Categories",
});
treedata.Add(new
{
 id = 11,
 pid=10,
 name = "Bestselling Albums",
});
treedata.Add(new
{
 id = 12,
 pid = 10,
 name = "New Releases"
});
treedata.Add(new
{
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
});
treedata.Add(new
{
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
});
treedata.Add(new
{
 id = 15,
 pid = 14,
 name = "Rock"
});
treedata.Add(new
{
 id = 16,
 name = "Gospel",
 pid = 14,
});
treedata.Add(new
{
 id = 17,
 pid = 14,
 name = "Latin Music"
});
treedata.Add(new
{
 id = 18,
 pid = 14,
 name = "Jazz"
});
treedata.Add(new
{
```

```
 id = 19,
 hasChild = true,
 name = "More in Music"
 });
 treedata.Add(new
 {
 id = 20,
 pid = 19,
 name = "Music Trade-In"
 });
 treedata.Add(new
 {
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
 });
 treedata.Add(new
 {
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
 });
 ViewBag.dataSource = treedata;
 List<object> menuItems = new List<object>();
 menuItems.Add(new
 {
 text = "Add New Item",
 });
 menuItems.Add(new
 {
 text = "Rename Item",
 });
 menuItems.Add(new
 {
 text = "Remove Item",
 });
 ViewBag.menuItems = menuItems;
 return View();
}
}
```

Output be like the below.



Check/uncheck the checkbox on clicking the tree node text

You can check and uncheck the checkboxes of tree view by clicking the tree node using the `nodeClicked` event of TreeView.

### CSHTML

```
@Html.EJS().TreeView("treedata").ShowCheckBox(true).NodeClicked("nodeCheck")
.KeyPress("nodeCheck").Fields(field=>

field.Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded
("expanded")

.Selected("is_selected").HtmlAttributes("hasAttribute").DataSource(ViewBag.d
ataSource)).Render()
<script>
 function nodeCheck(args) {
 var checkedNode = [args.node];
 var treeObj = document.getElementById('treedata').ej2_instances[0];
 if (args.event.target.classList.contains('e-fullrow') ||
args.event.key == "Enter") {
 var getNodeDetails = treeObj.getNode(args.node);
 if (getNodeDetails.isChecked == 'true') {
 treeObj.uncheckAll(checkedNode);
 } else {
 treeObj.checkAll(checkedNode);
 }
 }
 }
}
</script>
```

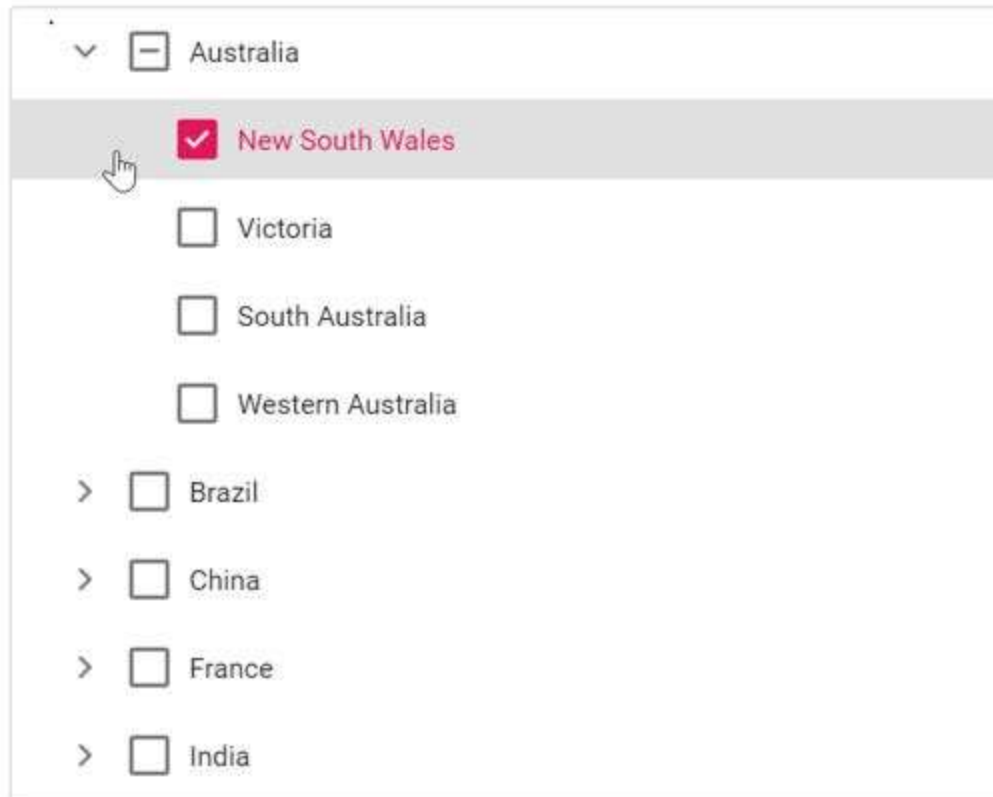
**NODECHECK.CS**

```
public IActionResult CheckBox()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Australia",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "New South Wales",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Victoria"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "South Australia"
 });
 treedata.Add(new
 {
 id = 6,
 pid = 1,
 name = "Western Australia",
 });
 treedata.Add(new
 {
 id = 7,
 name = "Brazil",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 8,
 pid = 7,
 name = "Paraná"
 });
 treedata.Add(new
 {
 id = 9,
 pid = 7,
 name = "Ceará"
 });
 treedata.Add(new
 {
 id = 10,
```

```
 pid = 7,
 name = "Acre"
 });
 treedata.Add(new
 {
 id = 11,
 name = "China",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 12,
 pid = 11,
 name = "Guangzhou"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 11,
 name = "Shanghai"
 });
 treedata.Add(new
 {
 id = 14,
 pid = 11,
 name = "Beijing"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 11,
 name = "Shantou"
 });
 treedata.Add(new
 {
 id = 16,
 name = "France",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 17,
 pid = 16,
 name = "Pays de la Loire"
 });
 treedata.Add(new
 {
 id = 18,
 pid = 16,
 name = "Aquitaine"
 });
 treedata.Add(new
 {
 id = 19,
 pid = 16,
 name = "Brittany"
 });
 });
```

```
 treedata.Add(new
 {
 id = 20,
 pid = 16,
 name = "Lorraine"
 });
 treedata.Add(new
 {
 id = 21,
 name = "India",
 hasChild = true
 });
 treedata.Add(new
 {
 id = 22,
 pid = 21,
 name = "Assam"
 });
 treedata.Add(new
 {
 id = 23,
 pid = 21,
 name = "Bihar"
 });
 treedata.Add(new
 {
 id = 24,
 pid = 21,
 name = "Tamil Nadu"
 });
 ViewBag.dataSource = treedata;
 return View();
 }
```

Output be like the below.



Validate the text when renaming the tree node

You can validate the tree node text while editing using `nodeEdited` event of the TreeView. Following is an example that shows how to validate and prevent empty values in tree node.

#### CSHTML

```
<div id='treeparent'>
 <div>

@Html.EJS().TreeView("treedata").NodeEdited("onNodeEdited").AllowEditing(true).Fields(field=>

field.Id("id").ParentID("pid").Text("name").HasChildren("hasChild").Expanded("expanded")
.DataSource(ViewBag.dataSource)).Render()

 </div>
</div>
<div id="display"></div>
<style>
 #treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
 }
 #display {
```

```

 max-width: 500px;
 margin: auto;
 padding: 10px;
 }
</style>
<script>
 function onNodeEdited(args) {
 var treeObj = document.getElementById('treedata').ej2_instances[0];
 var displayContent = "";
 if (args.newText.trim() == "") {
 args.cancel = true;
 displayContent = "TreeView item text should not be empty";
 }
 else if (args.newText != args.oldText) {
 displayContent = "TreeView item text edited successfully";
 } else {
 displayContent = "";
 }
 document.getElementById("display").innerHTML = displayContent;
 }
</script>

```

### NODEVALIDATE.CS

```

public IActionResult NodeEdit()
{
 List<object> treedata = new List<object>();
 treedata.Add(new
 {
 id = 1,
 name = "Discover Music",
 hasChild = true,
 expanded = true
 });
 treedata.Add(new
 {
 id = 2,
 pid = 1,
 name = "Hot Singles",
 });
 treedata.Add(new
 {
 id = 3,
 pid = 1,
 name = "Rising Artists"
 });
 treedata.Add(new
 {
 id = 4,
 pid = 1,
 name = "Live Music"
 });
 treedata.Add(new
 {
 id = 5,
 hasChild = true,

```

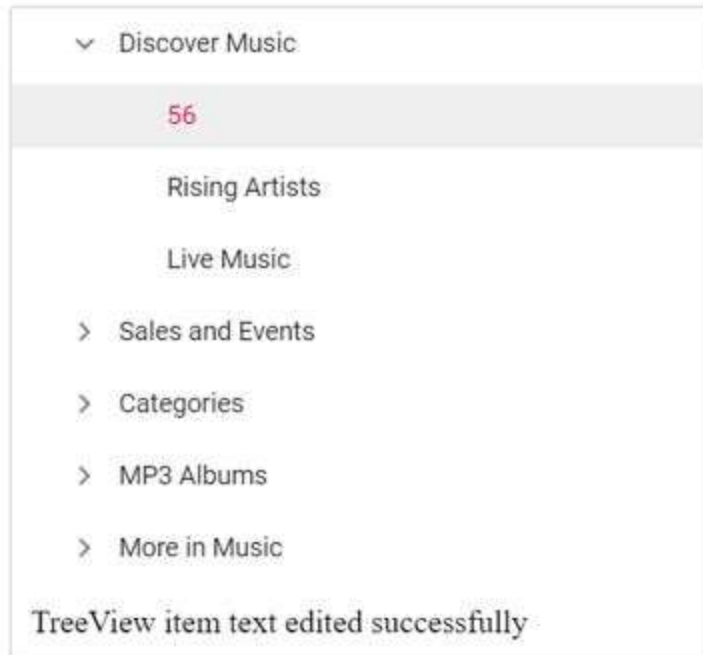


```
 name = "Sales and Events",
 });
 treedata.Add(new
 {
 id = 6,
 pid=5,
 name = "100 Albums - $5 Each",
 });
 treedata.Add(new
 {
 id = 7,
 pid = 5,
 name = "Hip-Hop and R&B Sale"
 });
 treedata.Add(new
 {
 id = 8,
 pid = 5,
 name = "CD Deals"
 });
 treedata.Add(new
 {
 id = 10,
 hasChild = true,
 name = "Categories"
 });
 treedata.Add(new
 {
 id = 11,
 pid=10,
 name = "Bestselling Albums",
 });
 treedata.Add(new
 {
 id = 12,
 pid = 10,
 name = "New Releases"
 });
 treedata.Add(new
 {
 id = 13,
 pid = 10,
 name = "Bestselling Songs"
 });
 treedata.Add(new
 {
 id = 14,
 hasChild = true,
 name = "MP3 Albums"
 });
 treedata.Add(new
 {
 id = 15,
 pid = 14,
 name = "Rock"
 });
});
```

```
treedata.Add(new
{
 id = 16,
 name = "Gospel",
 pid = 14,
});
treedata.Add(new
{
 id = 17,
 pid = 14,
 name = "Latin Music"
});
treedata.Add(new
{
 id = 18,
 pid = 14,
 name = "Jazz"
});
treedata.Add(new
{
 id = 19,
 hasChild = true,
 name = "More in Music"
});
treedata.Add(new
{
 id = 20,
 pid = 19,
 name = "Music Trade-In"
});
treedata.Add(new
{
 id = 21,
 name = "Redeem a Gift Card",
 pid = 19
});
treedata.Add(new
{
 id = 22,
 pid = 19,
 name = "Band T-Shirts"
});

ViewBag.dataSource = treedata;
return View();
}
```

Output be like the below.



### Customize the tree nodes based on levels

You can customize the tree nodes level wise by adding custom cssClass to the control and enabling styles.

#### CSHTML

```
<div id='treeparent'>
 @Html.EJS().TreeView("treedata").CssClass("mytree").Fields(field=>
field.Id("nodeId").ParentID("pid").Text("nodeText").HasChildren("hasChild").
Expanded("expanded")
 .Child(ViewBag.child).DataSource(ViewBag.dataSource)).Render()
 <label>Note:</label>
 <div>1. The font-weight "Bold" is applied for all the leaf
nodes</div>
 <div><i>2. The font-weight "Italic" is applied for first level
nodes</i></div>
 <div style="color: darkmagenta">3. The color "darkmagenta" is applied
for second level nodes</div>
</div>
<style>
 #treeparent {
 display: block;
 max-width: 350px;
 max-height: 350px;
 margin: auto;
 overflow: auto;
 border: 1px solid #dddddd;
 border-radius: 3px;
 }
 .details {
 padding-left: 10px;
 }
</style>
```

```

/*apply custom css to first level*/
.mytree .e-level-1 > .e-text-content .e-list-text {
 font-style: italic;
}
/*apply custom css to second level*/
.mytree .e-level-2 > .e-text-content .e-list-text {
 color: darkmagenta;
}
/*apply custom css to all the leaf nodes*/
.mytree .e-level-3 > .e-text-content .e-list-text {
 font-weight: bold;
}
</style>

```

### NODECUSTOMIZE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TreeView
{
 public partial class TreeViewController : Controller
 {
 public IActionResult DefaultFunctionalities()
 {
 List<Parentitem> parentitem = new List<Parentitem>();
 List<Childitem> childitem1 = new List<Childitem>();
 List<SubChilditem> subchilditem1 = new List<SubChilditem>();
 parentitem.Add(new Parentitem
 {
 nodeId = "01",
 nodeText = "Local Disk (C:)",
 expanded=true,
 child = childitem1,
 });
 childitem1.Add(new Childitem { nodeId = "01-01", nodeText =
"Program Files", child=subchilditem1 });
 subchilditem1.Add(new SubChilditem { nodeId = "01-01-01",
nodeText = "Windows NT" });
 subchilditem1.Add(new SubChilditem { nodeId = "01-01-02",
nodeText = "Windows Mail" });
 subchilditem1.Add(new SubChilditem { nodeId = "01-01-03",
nodeText = "Windows Photo Viewer" });
 List<SubChilditem> subchilditem2 = new List<SubChilditem>();
 childitem1.Add(new Childitem { nodeId = "01-02", nodeText =
"Users", expanded=true, child = subchilditem2 });
 subchilditem2.Add(new SubChilditem { nodeId = "01-02-01",
nodeText = "Smith" });
 subchilditem2.Add(new SubChilditem { nodeId = "01-02-02",
nodeText = "Public" });
 subchilditem2.Add(new SubChilditem { nodeId = "01-02-03",
nodeText = "Admin" });
 List<SubChilditem> subchilditem3 = new List<SubChilditem>();

```

```

 childitem1.Add(new Childitem { nodeId = "01-03", nodeText =
"Windows", child = subchilditem3 });
 subchilditem3.Add(new SubChilditem { nodeId = "01-03-01",
nodeText = "Boot" });
 subchilditem3.Add(new SubChilditem { nodeId = "01-03-02",
nodeText = "FileManager" });
 subchilditem3.Add(new SubChilditem { nodeId = "01-03-03",
nodeText = "System32" });
 List<Childitem> childitem2 = new List<Childitem>();
 parentitem.Add(new Parentitem
 {
 nodeId = "02",
 nodeText = "Local Disk (D:)",
 child = childitem2,
 });
 List<SubChilditem> subchilditem4 = new List<SubChilditem>();
 childitem2.Add(new Childitem { nodeId = "02-01", nodeText =
"Personals", child=subchilditem4});
 subchilditem4.Add(new SubChilditem { nodeId = "02-01-01",
nodeText = "My photo.png" });
 subchilditem4.Add(new SubChilditem { nodeId = "02-01-02",
nodeText = "Rental document.docx" });
 subchilditem4.Add(new SubChilditem { nodeId = "02-01-03",
nodeText = "Pay slip.pdf" });
 List<SubChilditem> subchilditem5 = new List<SubChilditem>();
 childitem2.Add(new Childitem { nodeId = "02-02", nodeText =
"Projects", child=subchilditem5 });
 subchilditem5.Add(new SubChilditem { nodeId = "02-02-01",
nodeText = "ASP Application " });
 subchilditem5.Add(new SubChilditem { nodeId = "02-02-02",
nodeText = "TypeScript Application" });
 subchilditem5.Add(new SubChilditem { nodeId = "02-02-03",
nodeText = "React Application" });

 List<SubChilditem> subchilditem6 = new List<SubChilditem>();
 childitem2.Add(new Childitem { nodeId = "02-02", nodeText =
"Office", child = subchilditem6 });
 subchilditem6.Add(new SubChilditem { nodeId = "02-03-01",
nodeText = "Work details.docx " });
 subchilditem6.Add(new SubChilditem { nodeId = "02-03-02",
nodeText = "Weekly report.docx" });
 subchilditem6.Add(new SubChilditem { nodeId = "02-03-03",
nodeText = "Wish list.csv" });
 List<Childitem> childitem3 = new List<Childitem>();
 parentitem.Add(new Parentitem
 {
 nodeId = "03",
 nodeText = "Local Disk (E:)",
 child = childitem3,
 });

 List<SubChilditem> subchilditem7 = new List<SubChilditem>();
 childitem3.Add(new Childitem { nodeId = "03-01", nodeText =
"Pictures", child=subchilditem7});
 subchilditem7.Add(new SubChilditem { nodeId = "03-01-01",
nodeText = "Wind.jpg " });

```

```

 subchilditem7.Add(new SubChilditem { nodeId = "03-01-02",
nodeText = "Stone.jpg" });
 subchilditem7.Add(new SubChilditem { nodeId = "03-01-03",
nodeText = "Home.jpg" });

 List<SubChilditem> subchilditem8 = new List<SubChilditem>();
 childitem3.Add(new Childitem { nodeId = "03-02", nodeText =
"Documents", icon = "docx", child=subchilditem8});
 subchilditem8.Add(new SubChilditem { nodeId = "03-02-01",
nodeText = "Environment Pollution.docx " });
 subchilditem8.Add(new SubChilditem { nodeId = "03-02-02",
nodeText = "Global Warming.ppt" });
 subchilditem8.Add(new SubChilditem { nodeId = "03-02-03",
nodeText = "Social Network.pdf" });

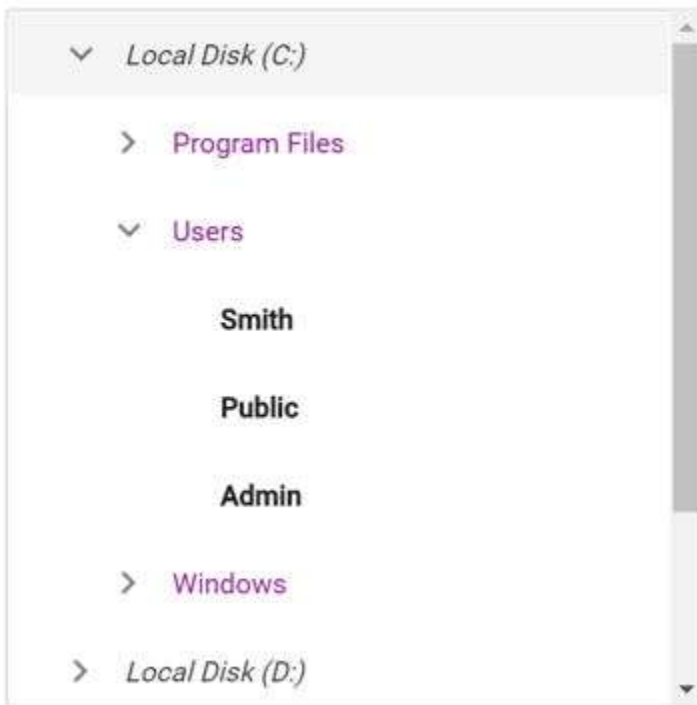
 List<SubChilditem> subchilditem9 = new List<SubChilditem>();
 childitem3.Add(new Childitem { nodeId = "03-03", nodeText =
"Study Materials",child=subchilditem9 });
 subchilditem9.Add(new SubChilditem { nodeId = "03-03-01",
nodeText = "UI-Guide.pdf" });
 subchilditem9.Add(new SubChilditem { nodeId = "03-03-02",
nodeText = "Tutorials.zip" });
 subchilditem9.Add(new SubChilditem { nodeId = "03-03-03",
nodeText = "TypeScript.7z" });
 ViewBag.dataSource = parentitem;
 char[] value = { 'c', 'h', 'i', 'l', 'd' };
 string Child = new string(value);
 ViewBag.child = Child;
 return View();
 }
}
}
}
public class Parentitem
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<Childitem> child;
}
public class Childitem
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<SubChilditem> child;
}
public class SubChilditem
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
}

```

```
}

```

Output be like the below.



### Restrict the drag-and-drop for particular tree nodes

You can able to restrict to drag and drop files under folder only. These can be achieved by using 'nodeDragStop' and 'nodeDragging' event of TreeView.

### CSHTML

```
@Html.EJS().TreeView("treedata").SortOrder(Syncfusion.EJ2.Navigations.SortOrder.Ascending).AllowDragAndDrop(true).NodeDragging("nodeDrag").NodeDragStop("dragStop").Fields(field=>
 field.Id("nodeId").Text("nodeText").HasChildren("hasChild").Expanded("expanded").IconCss("icon").ImageUrl("image")
 .Child(ViewBag.child).DataSource(ViewBag.dataSource)).Render()
<script>
 function nodeDrag(args) {
 if (args.droppedNode != null &&
 args.droppedNode.getElementsByClassName('folder') &&
 args.droppedNode.getElementsByClassName('folder').length === 0) {
 args.dropIndicator = 'e-no-drop';
 }
 }
 function dragStop(args) {
 if (args.droppedNode != null &&
 args.droppedNode.getElementsByClassName('folder') &&
 args.droppedNode.getElementsByClassName('folder').length === 0) {
 args.cancel = true;
 }
 }

```

```

 }
</script>
<style>
 .e-treeview .e-list-img {
 width: 25px;
 height: 25px;
 }
 /* Loading sprite image for TreeView */
 .e-treeview .e-list-icon {
 background-repeat: no-repeat;
 background-image:
url(https://ej2.syncfusion.com/demos/src/treeview/images/icons/file_icons.png);
 height: 20px;
 }
 /* Specify the icon positions based upon class name */
 .e-treeview .e-list-icon.folder {
 background-position: -10px -552px;
 }
 .e-treeview .e-list-icon.docx {
 background-position: -10px -20px;
 }
 .e-treeview .e-list-icon.ppt {
 background-position: -10px -48px;
 }
 .e-treeview .e-list-icon.pdf {
 background-position: -10px -104px;
 }
 .e-treeview .e-list-icon.images {
 background-position: -10px -132px;
 }
 .e-treeview .e-list-icon.zip {
 background-position: -10px -188px;
 }
 .e-treeview .e-list-icon.audio {
 background-position: -10px -244px;
 }
 .e-treeview .e-list-icon.video {
 background-position: -10px -272px;
 }
 .e-treeview .e-list-icon.exe {
 background-position: -10px -412px;
 }
</style>

```

### **NODEDRAG.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
// For more information on enabling MVC for empty projects, visit
https://go.microsoft.com/fwlink/?LinkID=397860
namespace EJ2CoreSampleBrowser.Controllers.TreeView
{

```



```

public partial class TreeViewController : Controller
{
 public IActionResult IconsandImages()
 {
 List<Parentitems> parentitem = new List<Parentitems>();
 List<Childitems> childitem = new List<Childitems>();
 parentitem.Add(new Parentitems
 {
 nodeId = "01",
 nodeText = "Music",
 icon="folder",
 child = childitem,
 });
 childitem.Add(new Childitems { nodeId = "01-01", nodeText =
"Gouttes.mp3", icon= "audio" });

 List<Childitems> childitem2 = new List<Childitems>();
 parentitem.Add(new Parentitems
 {
 nodeId = "02",
 nodeText = "Videos",
 icon = "folder",
 child = childitem2,
 });
 childitem2.Add(new Childitems { nodeId = "02-01", nodeText =
"Naturals.mp4", icon = "video" });
 childitem2.Add(new Childitems { nodeId = "02-02", nodeText =
"Wild.mpeg", icon = "video" });
 List<Childitems> childitem3 = new List<Childitems>();
 parentitem.Add(new Parentitems
 {
 nodeId = "03",
 nodeText = "Documents",
 icon = "folder",
 child = childitem3,
 });
 childitem3.Add(new Childitems { nodeId = "03-01", nodeText =
"Environment Pollution.docx", icon = "docx" });
 childitem3.Add(new Childitems { nodeId = "03-02", nodeText = "Global
Water, Sanitation, & Hygiene.docx", icon = "docx" });
 childitem3.Add(new Childitems { nodeId = "03-03", nodeText = "Global
Warming.ppt", icon = "ppt" });
 childitem3.Add(new Childitems { nodeId = "03-04", nodeText = "Social
Network.pdf", icon = "pdf" });
 childitem3.Add(new Childitems { nodeId = "03-05", nodeText = "Youth
Empowerment.pdf", icon = "pdf" });
 childitem2.Add(new Childitems { nodeId = "02-01", nodeText =
"Naturals.mp4", icon = "video" });
 childitem2.Add(new Childitems { nodeId = "02-02", nodeText =
"Wild.mpeg", icon = "video" });
 List<Childitems> childitem4 = new List<Childitems>();
 parentitem.Add(new Parentitems
 {
 nodeId = "04",
 nodeText = "Downloads",
 icon = "folder",

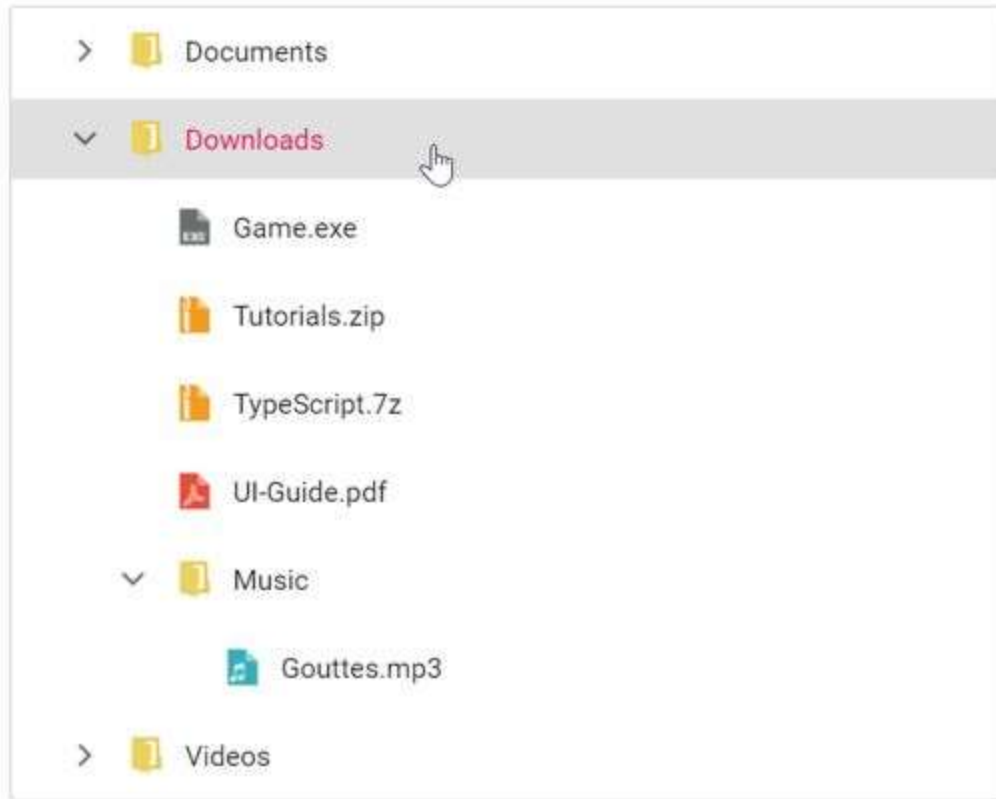
```

```

 child = childitem4,
 });
 childitem4.Add(new Childitems { nodeId = "04-01", nodeText = "UI-
Guide.pdf", icon = "pdf" });
 childitem4.Add(new Childitems { nodeId = "04-02", nodeText =
"Tutorials.zip", icon = "zip" });
 childitem4.Add(new Childitems { nodeId = "04-03", nodeText =
"Game.exe", icon = "exe" });
 childitem4.Add(new Childitems { nodeId = "04-04", nodeText =
"TypeScript.7z", icon = "zip" });
 ViewBag.dataSource = parentitem;
 char[] value = { 'c', 'h', 'i', 'l', 'd' };
 string Child = new string(value);
 ViewBag.child = Child;
 return View();
}
}
}
public class Parentitems
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
 public List<Childitems> child;
}
public class Childitems
{
 public string nodeId;
 public string nodeText;
 public string icon;
 public bool expanded;
 public bool selected;
}
}

```

Output be like the below.



## Migration from Essential JS 1

This article describes the API migration process of TreeView component from Essential JS 1 to Essential JS 2.

### Add nodes

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Add node | **Method:** `addNode`  
`<@Html.EJ().TreeView("tree").TreeViewFields(field =>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources).Id("Id").ParentId("Parent").Text("Text"))>`  
`var treeObj = $("#tree").data("ejTreeView");`  
`treeObj.addNode("Node", "#book");` | **Method:** `addNodes`  
`<@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()>`  
`<Script>`  
`var treeObj = document.getElementById('tree').ej2_instances[0];`  
`var object = [{ id: "temp", name: "New node" }, { id: "new", name: "New node 1" }];`  
`treeObj.addNodes(object, "book");` |

| Triggers before adding node | **Event:** `BeforeAdd`  
`<@Html.EJ().TreeView("tree").TreeViewFields(field =>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources).Id("Id").ParentId("Parent").Text("Text"))>`  
`.ClientSideEvents(events =>events.BeforeAdd("beforeAdd"))>`  
`<Script>`  
`function beforeAdd() {}` | Not Applicable |

| Adding node after a particular node | **Method:** *insertAfter*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br /><br /></Script><br />var treeObj =  
 \$("#tree").data("ejTreeView");<br />treeObj.insertAfter("node", "book"); | **Can be achieved  
 using**<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()<br />var object =  
 [{ id: "1", name: "node" }];<br />var treeObj =  
 document.getElementById('tree').ej2\_instances[0]<br />var child =  
 treeObj.element.querySelector('[data-uid="book"]');<br />var parent =  
 child.parentElement.closest('.e-list-item');<br />var level = parseInt(parent.getAttribute('aria-  
 level'))+1;<br />var childNodes = Array.from(parent.querySelectorAll('.e-list-item.e-level-  
 '+level))<br />var index = childNodes.indexOf(child)<br />treeObj.addNodes(object, "book",  
 index+1); |

| Adding node before a particular node | **Method:** *insertBefore*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br /><br /></Script><br />var treeObj =  
 \$("#tree").data("ejTreeView");<br />treeObj.insertBefore("node", "book"); | **Can be achieved  
 using**<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()<br />var object =  
 [{ id: "1", name: "node" }];<br />var treeObj =  
 document.getElementById('tree').ej2\_instances[0]<br />var child =  
 treeObj.element.querySelector('[data-uid="book"]');<br />var parent =  
 child.parentElement.closest('.e-list-item');<br />var level = parseInt(parent.getAttribute('aria-  
 level'))+1;<br />var childNodes = Array.from(parent.querySelectorAll('.e-list-item.e-level-  
 '+level))<br />var index = childNodes.indexOf(child)<br />treeObj.addNodes(object, "book",  
 index-1); |

| Triggers when node is added successfully | **Event:** *NodeAdd*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text")).ClientSideEvents(events  
 =>events.NodeAdd("nodeAdd"))<br /><br /></Script><br />function nodeAdd() {} | **Event:  
 DataSourceChanged**<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).DataSourceChanged("DataSourceChang  
 ed").Render()<br /><br /></Script><br /><br />function DataSourceChanged(args){ |

## Common

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Keyboard Navigation | **Property:** *AllowKeyboardNavigation*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text")).AllowKeyboardNavigation(false)) | **Can be achieved  
 using**,<br /><br />

```

/>@Html.EJS().TreeView("tree").Fields(ViewBag.TemplateFields).KeyPress("onKeyPress").Render()

</Script>

function onKeyPress(args){
args.cancel = true
} |

```

| Triggers before node is cut | **Event:** *BeforeCut*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.BeforeCut("beforeCut"))
function beforeCut() {} | Not Applicable |

```

| Triggers before node is deleted | **Event:** *BeforeDelete*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.BeforeDelete("beforeDelete"))
function beforeDelete() {} | Not Applicable |

```

| Triggers before loading nodes | **Event:** *BeforeLoad*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.BeforeLoad("beforeLoad"))
function beforeLoad() {} | Not Applicable |

```

| Triggers before node is pasted | **Event:** *BeforePaste*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.BeforePaste("beforePaste"))
function beforePaste() {} | Not Applicable |

```

| Triggers when Treeview is created | **Event:** *Create*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.Create("create"))
function create() {} | Event: Created

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Created("onCreated").Render()

/></Script>

function onCreated(args){} |

```

| Css class | **Property:** *CssClass*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).CssClass("custom")) | Property: CssClass

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).CssClass("custom").Render() |

```

| Triggers when Treeview is destroyed | **Event:** *Destroy*<br /><br />  

```

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.Destroy("destroy"))
function destroy() {} | Event: Destroyed

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Destroyed("onDestroy").Render()

/>
</Script>

function onDestroy(args){} |

```

```

| Destroy Treeview control | Method: destroy

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.DataSource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.destroy(); | Method: destroy

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var
treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.destroy(); |

| Disable Node | Method: disableNode

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.DataSource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.disableNode($("#1")); | Method: disableNodes

/>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

/>var treeObj = document.getElementById('tree').ej2_instances[0]

/>treeObj.disableNodes(["1", "2"]); |

| Enable Animation | Property: EnableAnimation

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.DataSource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).EnableAnimation(false)) | Property: Animation

/>

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Animation(ViewBag.TreeViewNodeAni
mation).Render()

</Controller>

TreeViewNodeAnimationSettings Animation =
new TreeViewNodeAnimationSettings();
List<Object> expand = new List<object>();

/>expand.Add(new{
duration = 0
});
Animation.Expand = expand[0];

/>List<Object> collapse = new List<object>();
collapse.Add(new{
duration = 0

/>});
Animation.Collapse = collapse[0];
ViewBag.TreeViewNodeAnimation =
Animation; |

| Control state | Property: Enabled

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.DataSource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).Enabled(false)) | Not Applicable |

| Enable Node | Method: enableNode

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.DataSource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.enableNode($("#1")); | Method: enableNodes

/>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

/>var treeObj = document.getElementById('tree').ej2_instances[0]

/>treeObj.enableNodes(["1"]); |

| Persistence | Property: EnablePersistence

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.DataSource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).EnablePersistence(true)) | Property:

```

```

EnablePersistence

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).EnablePersistence(true).Render() |
| Right to Left | Property: EnableRTL

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).EnableRTL(true)) | Property: EnableRtl

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).EnableRtl(true).Render() |

| Ensure visibility | Method: ensureVisible

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.ensureVisible($("#1")); | Method: ensureVisible

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.ensureVisible("1"); |

| Mapping fields | Property: TreeViewFields

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).Child("Child").HasChild("HasChild")
.Expanded("Expanded").HtmlAttribute("HtmlAttributes").ImageAttribute("img").ImageUrl("Im
ageUrl").IsChecked("Checked")
.LinkAttribute("LinkAttribute").Query(null).Selected("Selected").SpriteCssClass("custom").Tab
leName(null)) | Property: Fields

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields)

</Controller>
TreeViewFieldsSettings TemplateFields = new TreeViewFieldsSettings();
TreeViewTemplate template = new TreeViewTemplate();
TemplateFields.DataSource =
template.getTreeViewTemplate();
TemplateFields.IsChecked = "checked"
TemplateFields.Id = "id";
TemplateFields.Text = "name";
TemplateFields.ParentID =
"pid";
TemplateFields.HasChildren = "HasChild";
TemplateFields.Expanded =
"Expanded";
TemplateFields.Selected = "Selected";
ViewBag.TemplateFields =
TemplateFields; |

| Get child nodes | Method: getChildren

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getChildren("1"); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields)

</Script>
var instance=
document.getElementById("tree").ej2_instances[0];
var parent =
instance.element.querySelector('[data-uid="1"]')
console.log(parent.querySelector('.e-list-
item')) |

| Get node | Method: getNode

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =

```

```
$("#tree").data("ejTreeView");
treeObj.getNode($("#1")); | Method: getNode

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.getNode("1"); |
```

```
| Get node by index | Method: getNodeByIndex @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) </Script> var treeObj = $("#tree").data("ejTreeView"); treeObj.getNodeByIndex(3); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var instance= document.getElementById("tree").ej2_instances[0]; var nodes=instance.element.querySelectorAll('.e-list-item') console.log(nodes[3]); |
```

```
| Get node count | Method: getNodeCount @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) </Script> var treeObj = $("#tree").data("ejTreeView"); treeObj.getNodeCount(); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var instance= document.getElementById("tree").ej2_instances[0]; var nodes = instance.element.querySelectorAll('.e-list-item') console.log("Node count is " + nodes.length); |
```

```
| Get node index | Method: getNodeIndex

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj = $("#tree").data("ejTreeView");
treeObj.getNodeIndex($("#book")); | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var instance= document.getElementById("tree").ej2_instances[0];
var nodes=instance.element.querySelectorAll('.e-list-item')
var node = instance.element.querySelector('[data-uid="'+ "book" + "']');
while(i<nodes.length) {
if(nodes[i] === node) {console.log("Node index is "+i)
break;
}
i++
}
}
```

```
| Get parent of node | Method: getParent @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) </Script> var treeObj = $("#tree").data("ejTreeView"); treeObj.getParent($("#book")); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var instance= document.getElementById("tree").ej2_instances[0]; var child=instance.element.querySelector('[data-uid="'+ "book" + "']'); var parent = child.parentNode.closest('.e-list-item') console.log(parent) |
```

```
| Get tree node text | Method: getText

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

```



```

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getText("1"); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var
instance= document.getElementById("tree").ej2_instances[0];
var
nodeText=instance.getNode("1")['text'] |

| Get updated datasource | Method: getTreeData

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getTreeData(); | Method: getTreeData

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var
treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.getTreeData(); |

| Get visible nodes | Method: getVisibleNodes

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getVisibleNodes(); | Not Applicable |

| Height of Treeview control | Property: Height

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).Height("400px") | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).CssClass("custom").Render()
</Css>

/>.e-treeview.custom{
height: 400px;
} |

| Checking for child nodes | Method: hasChildNode

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.hasChildNode("book"); | Can be achieved using,

/>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var
instance= document.getElementById("tree").ej2_instances[0];
var
parent=instance.element.querySelector('[data-uid="book"]')
if (parent.querySelector('.e-
list-item') !== null) {
console.log("Has child node")
} |

| Hide all nodes | Method: hide

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.hide(); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var
instance= document.getElementById("tree").ej2_instances[0];

/>instance.element.querySelector('.e-list-parent').style.display="none" |

| Hide node | Method: hideNode

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =

```

```
$(("#tree").data("ejTreeView"));
treeObj.hideNode("book"); | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var instance= document.getElementById("tree").ej2_instances[0];
instance.element.querySelector('[data-uid="book"]').style.display="none" |
```

```
| HTML Attributes | Property: HtmlAttributes @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")).HtmlAttributes(htmlAttr) </Script> @{ Dictionary<string, object> htmlAttr = new Dictionary<string, object>(); htmlAttr.Add("name", "treeView"); } | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var element= document.getElementById("tree"); element.classList.add("htmlAttr") |
```

```
| To check if child nodes are loaded | Method: isChildLoaded @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) </Script> var treeObj = $(("#tree").data("ejTreeView")); treeObj.isChildLoaded("book"); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var instance= document.getElementById("tree").ej2_instances[0]; var parent=instance.element.querySelector('[data-uid="book"]') if (parent.querySelector('.e-list-item') !== null) { console.log("Child is loaded") } |
```

```
| To check if node is disabled | Method: isDisabled @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) </Script> var treeObj = $(("#tree").data("ejTreeView")); treeObj.isDisabled("book"); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var instance= document.getElementById("tree").ej2_instances[0]; var node=instance.element.querySelector('[data-uid="book"]') if (node.classList.contains('e-disable') === true) { console.log("Node is disabled") } |
```

```
| To check if node exists in Treeview | Method: isExist @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) </Script> var treeObj = $(("#tree").data("ejTreeView")); treeObj.isExist("book"); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() </Script> var instance= document.getElementById("tree").ej2_instances[0]; if (instance.getNode('book')['text'] !== "") { console.log("Node exists") } |
```

```
| To check if node is visible | Method: isVisible

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
```

```
$(("#tree").data("ejTreeView"));
treeObj.isVisible("book"); | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var instance= document.getElementById("tree").ej2_instances[0];
if (instance.element.querySelector('[data-uid="book"]').style.display !== "none"){
console.log("Node is visible")
}
```

```
| Triggers on key press | Event: KeyPress @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasource) .Id("Id").ParentId("Parent").Text("Text")) .ClientSideEvents(events =>events.KeyPress("keyPress")) function keyPress() {} | Event: KeyPress @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).KeyPress("onKeyPress").Render() Script function onKeyPress(){} |
```

```
| Load Treeview nodes from particular URL | Method: loadData @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasource) .Id("Id").ParentId("Parent").Text("Text")) Script var treeObj = $(("#tree").data("ejTreeView")); treeObj.loadData("childData", "book"); | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() Script var instance= document.getElementById("tree").ej2_instances[0]; TreeViewFieldsSettings data= new TreeViewFieldsSettings(); object dataManager = new DataManager { Url = "/FileContent/rootNode", Adaptor = "UrlAdaptor", CrossDomain = true}; dataManager.executeQuery(new ej.data.Query().take(8).then((e) => { var childData = e.result; instance.addNodes(childData, "book") })); |
```

```
| Triggers when data load fails | Event: LoadError @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasource) .Id("Id").ParentId("Parent").Text("Text")) .ClientSideEvents(events =>events.LoadError("loadError")) function loadError() {} | Can be achieved using, @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render() Script var instance= document.getElementById("tree").ej2_instances[0]; TreeViewFieldsSettings data= new TreeViewFieldsSettings(); object dataManager = new DataManager { Url = "/FileContent/rootNode", Adaptor = "UrlAdaptor", CrossDomain = true}; dataManager.executeQuery(new ej.data.Query().take(8).error((e) => { console.log('Data loaded failed') })); |
```

```
| Load on demand | Property: LoadOnDemand @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasource) .Id("Id").ParentId("Parent").Text("Text")).LoadOnDemand(true) | Property: LoadOnDemand @Html.EJS().TreeView("tree").Fields(ViewBag.Fields).LoadOnDemand(true).Render()Treeview is rendered in load on demand by default |
```

| Triggers when data load is success | **Event:** *LoadSuccess*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events  
 =>events.LoadSuccess("loadSuccess"))<br />function loadSuccess() {}<br />var treeObj =  
 document.getElementById('tree').ej2instances[0]<br />treeObj.loadData("childData", "book"); |  
 Can be achieved using,<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()<br /><br />Script<br />var  
 instance= document.getElementById("tree").ej2instances[0];<br />TreeViewFieldsSettings data=  
 new TreeViewFieldsSettings();<br />object dataManager = new DataManager {<br />Url =  
 "/FileContent/rootNode",<br />Adaptor = "UrlAdaptor",<br />CrossDomain = true};<br />  
 />dataManager.executeQuery(new ej.data.Query().take(8).then((e) => {<br />console.log('Data  
 loaded successfully')<br />}}); |

| To move node | **Method:** *moveNode*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br /><br />Script<br />var treeObj =  
 \$("#tree").data("ejTreeView");<br />treeObj.moveNode("book", "art"); | **Method:** *moveNodes*<br />  
 /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()<br /><br />Script<br /><br />  
 />var treeObj = document.getElementById('tree').ej2\_instances[0]<br />  
 />treeObj.moveNodes("book", "art"); |

| Triggers when node is clicked successfully | **Event:** *NodeClick*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events  
 =>events.NodeClick("nodeClick"))<br />function nodeClick() {} | **Event:** *NodeClicked*<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeClicked("onNodeClick").Render()<br />  
 /><br />Script<br /><br />function onNodeClick(){} |

| Triggers when node is cut successfully | **Event:** *NodeCut*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events  
 =>events.NodeCut("nodeCut"))<br />function nodeCut() {} | Not Applicable |

| Triggers when node is deleted successfully | **Event:** *NodeDelete*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events  
 =>events.NodeDelete("nodeDelete"))<br />function nodeDelete() {} | **Event:**  
*DataSourceChanged*<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).DataSourceChanged("DataSourceChang  
 ed").Render()<br /><br />Script<br /><br />function DataSourceChanged(args){ |

```
| Triggers when node is pasted successfully | Event: NodePaste @Html.EJ().TreeView("tree").TreeViewFields(field => field.Datasource((IEnumerable<LoadData>)ViewBag.datasources) .Id("Id").ParentId("Parent").Text("Text")) .ClientSideEvents(events =>events.NodePaste("nodePaste")) function nodePaste() {} | Not Applicable |
```

```
| Triggers when nodes are loaded successfully | Event: Ready

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.Ready("onReady"))
function onReady() {} | Event: DataBound

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).DataBound("onDataBound").Render()

</script>

function onDataBound(){} |
```

```
| Refresh Treeview control | Method: refresh

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.refresh(); | Not Applicable |
```

```
| To show all nodes | Method: show

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br
</>.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.show(); | Can be achieved using
<br
</>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var
instance= document.getElementById("tree").ej2_instances[0];<br
</>instance.element.querySelector('e-list-parent').style.display="block" |
```

```
| Show node | Method: showNode

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br

.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.showNode("book"); | Can be achieved using,<br

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var
instance= document.getElementById("tree").ej2_instances[0];<br

instance.element.querySelector("[data-uid="book"]').style.display="block" |
```

```
| Remove all Treeview nodes | Method: removeAll

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.removeAll(); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var
instance= document.getElementById("tree").ej2_instances[0];

/>instance.removeNodes([instance.element.querySelector('.e-list-parent')]) |
```

```
| Remove Treeview node | Method: removeNode

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)<br
```

```

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.removeNode("book"); | Method: removeNodes

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.removeNodes("book"); |

| Sort order | Property: SortSettings

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).SortSettings(s =>
s.AllowSorting(true).SortOrder(SortOrder.Descending)) | Property: SortOrder

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).SortOrder(SortOrder.Descending).Rend
er() |

| Update node text | Method: updateText

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.updateText("book", "text"); | Method:
updateNode

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.updateNode("book", "text"); |

| Width of Treeview control | Property: Width

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).Width("300px") | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).CssClass("custom").Render()
</Css>
.e-treeview.custom{
width: 300px;
} |

```

## CheckBox

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```

| Prevent auto-check of child and parent | Property: AutoCheck

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true).AutoCheck(false) | Property:
AutoCheck

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AutoCheck(false).ShowCheckBox(true).R
ender() |

| Prevent indeterminate state in parent node | Property: AutoCheckParentNode

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true).AutoCheckParentNode(true) |
Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).AutoCheck(false).
NodeChecked("nodeChecked").Render()

</Script>
var treeObj=

```

```
document.getElementById("tree").ej2_instances[0];
function nodeChecked(args){
var
child = treeObj.element.querySelector("[data-uid='"+ args.data[0]['id'] + "'");
var
checkNodes = [];
var element = child.parentNode;
while ((element !== null || element
!== undefined) && !element.parentNode.classList.contains('e-treeview')) {
element =
element.parentNode;
var id = element.getAttribute('data-uid');
if (id !== null)
checkNodes.push(element.getAttribute('data-uid'))
}
if (child.querySelector('.e-list-
item') !== null && args.isInteracted === true && args.action === 'check')
{
treeObj.autoCheck = true;
treeObj.checkAll(child.getAttribute('data-uid'));
} else
if (child.querySelector('.e-list-item') !== null && args.isInteracted === true && args.action ===
'uncheck') {
treeObj.autoCheck = true;
treeObj.uncheckAll(child.getAttribute('data-
uid'));
}
treeObj.autoCheck = false;
if (args.action === 'check')
{
treeObj.checkAll(checkNodes)
}
else if (args.action === 'uncheck' &&
child.parentNode.querySelector('.e-check') === null)
{
treeObj.uncheckAll(checkNodes)
}
}
```

```
| Check all nodes | Method: checkAll

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))).ShowCheckbox(true)

Script
var
treeObj = $("#tree").data("ejTreeView");
treeObj.checkAll(); | Method: checkAll

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).Render()

Script

var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.checkAll(); |
```

```
| Check node | Method: checkNode

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))).ShowCheckbox(true)

Script
var
treeObj = $("#tree").data("ejTreeView");
treeObj.checkNode("book"); | Method:
checkAll

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).Render()

Script

var treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.checkAll("book"); |
```

```
| Set checkednodes | Property: CheckedNodes

List<int> checkedNodes = new
List<int>();
checkedNodes.Add(2);
checkedNodes.Add(8);
checkedNodes.Add(12);
@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true).CheckedNodes(ViewBag.check
ednodes) | Property: CheckedNodes

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).CheckedNodes(Vie
wBag.CheckedNodes).Render()

Controller

ViewBag.CheckedNodes = new
string[] { "3", "9", "13" }; |
```

```
| Get checked nodes | Method: getCheckedNodes

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))).ShowCheckbox(true)

Script
var
```

```

treeObj = $("#tree").data("ejTreeView");
treeObj.getCheckedNodes(); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).Render()

/>Script
var instance= document.getElementById("tree").ej2_instances[0];
var check=instance.getAllCheckedNodes();
var i=0;
var checkedNodes;

/>while(i<check.length) {
checkedNodes.push(instance.element.querySelector('[data-uid="" + checkedNodes[i] + ""']));
i++
}
console.log(checkedNodes) |

| Get checked nodes index | Method: getCheckedNodesIndex

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true)

Script
var treeObj = $("#tree").data("ejTreeView");
treeObj.getCheckedNodesIndex(); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).CheckedNodes("1").Render()

Script
var instance= document.getElementById("tree").ej2_instances[0];
var nodes = instance.element.querySelectorAll('e-list-item')
var nodeIndex[];

/>while(i<nodes.length) {
if(nodes[i].classList.contains('e-check')) {nodeIndex.push(i);
}
i++
}
console.log(nodeIndex) |

| To check if nodes are checked | Method: isNodeChecked

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true)

Script
var treeObj = $("#tree").data("ejTreeView");
treeObj.isNodeChecked("book"); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).Render()

Script
var instance= document.getElementById("tree").ej2_instances[0];
var checked=instance.getNode("book")['isChecked']
if (checked === true) {
console.log("Node is checked")
} |

| Triggers when node is checked successfully | Event: NodeCheck

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true)
.ClientSideEvents(events =>events.NodeCheck("nodeCheck"))
function nodeCheck() {} | Event: nodeChecked

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).NodeChecked("nodeChecked").Render()

Script

function nodeChecked(args){
if (args.action == "check") {}
} |

| Checkbox support | Property: ShowCheckbox

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true).ShowCheckbox(true) | Property: ShowCheckBox

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true) |

```



| To uncheck all nodes | **Method:** *unCheckAll*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true)<br /><br /></Script><br /><br />  
 treeObj = \$("#tree").data("ejTreeView");<br />treeObj.unCheckAll(); | **Method:** *uncheckAll*<br />  
 /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).Render()<br />  
 /><br /></Script><br /><br />var treeObj = document.getElementById('tree').ej2\_instances[0]<br />  
 />treeObj.uncheckAll(); |

| To uncheck node | **Method:** *uncheckNode*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true)<br /><br /></Script><br /><br />  
 treeObj = \$("#tree").data("ejTreeView");<br />treeObj.uncheckNode("book"); | **Method:** *uncheckAll*<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).Render()<br /><br />  
 /></Script><br /><br />var treeObj = document.getElementById('tree').ej2\_instances[0]<br />  
 />treeObj.uncheckAll("book"); |

| Triggers when node is unchecked successfully | **Event:** *NodeUncheck*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text")).ShowCheckbox(true)<br /><br />.ClientSideEvents(events  
 =>events.NodeUncheck("nodeUncheck"))<br />function nodeUncheck() {} | **Event:** *nodeChecked*<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).NodeChecked("no  
 deChecked").Render()<br /><br /></Script><br /><br />function nodeChecked(args){<br />if  
 (args.action == "un-check") {}<br />} |

| Triggers before nodes are checked/ unchecked | Not Applicable | **Event:** *NodeChecking*<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ShowCheckBox(true).NodeChecking("no  
 deChecking").Render()<br /><br /></Script><br /><br />function nodeChecking(args){<br />if  
 (args.action == "check") {}<br />} |

## Drag and Drop

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Drag and drop | **Property:** *AllowDragAndDrop*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 />.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true) | **Property:** *AllowDragAndDrop*<br /><br />  
 />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true) |

| Prevent Drag and drop to another Treeview | **Property:** *AllowDragAndDropAcrossControl*<br /><br />  
 />@Html.EJ().TreeView("tree").TreeViewFields(field =><br />  
 />field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />

```

/>.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true).AllowDragAndDropAcross
Control(false) | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDragStop
("dragStop").Render()

</Script>

function dragStop(args){
if
(args.draggedParentNode.closest('.e-treeview') !== args.dropTarget.closest('.e-treeview')) {
args.cancel = true;
}
} |

| Prevent sibling drop | Property: AllowDropSibling

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true).AllowDropSibling(false) |
Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDragStop
("dragStop").Render()

</Script>

function dragStop(args){
if(args.dropIndicator === "e-drop-next") {
args.cancel = true;
}
} |

| Prevent child drop | Property: AllowDropChild

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true).AllowDropChild(false) |
Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDragStop
("dragStop").Render()

</Script>

function dragStop(args){
if(args.dropIndicator === "e-drop-in") {
args.cancel = true;
}
} |

| Triggers when node is dragged | Event: NodeDrag

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true)
.ClientSideEvents(events =>events.NodeDrag("nodeDrag"))
function nodeDrag() {} |
Event: NodeDragging

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDragging
("nodeDrag").Render()

</Script>

function nodeDrag(){ } |

| Triggers when node drag is started successfully | Event: NodeDragStart

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true)
.ClientSideEvents(events =>events.NodeDragStart("nodeDragStart"))
function
nodeDragStart() {} | Event: NodeDragStart

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDragStar
t("nodeDragStart").Render()

</Script>

function nodeDragStart(){ } |

| Triggers before dragged node drag is stopped | Event: NodeDragStop

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true)<br

```

```

/>.ClientSideEvents(events =>events.NodeDragStop("nodeDragStop"))
function
nodeDragStop() {} | Event: NodeDragStop

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDragStop
("nodeDragStop").Render()

</Script>

function nodeDragStop(){} |
| Triggers when node is dropped successfully | Event: NodeDropped

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text")).AllowDragAndDrop(true)

/>.ClientSideEvents(events =>events.NodeDropped("nodeDropped"))
function
nodeDropped() {} | Event: NodeDropped

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowDragAndDrop(true).NodeDropped
("nodeDropped").Render()

</Script>

function nodeDropped(){} |

```

### Expand/Collapse nodes

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```

| Triggers before node is collapsed | Event: BeforeCollapse

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

/>.ClientSideEvents(events =>events.BeforeCollapse("beforeCollapse"))
function beforeCollapse() {} | Event:
NodeCollapsing

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeCollapsing("nodeCollapsing").Render
()

</Script>

function nodeCollapsing(){} |

```

```

| Triggers before node is expanded | Event: BeforeExpand

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

/>.ClientSideEvents(events =>events.BeforeExpand("beforeExpand"))
function beforeExpand() {} | Event:
NodeExpanding

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeExpanding("nodeExpanding").Render
()

</Script>

function nodeExpanding(){} |

```

```

| Collapse all nodes | Method: collapseAll

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.collapseAll(); | Method: collapseAll

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var
treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.collapseAll(); |

```

```

| Collapse Node | Method: collapseNode

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =

```

```

$(("#tree").data("ejTreeView"));
treeObj.collapseNode("book"); | Method: collapseAll

/>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

/>var treeObj = document.getElementById('tree').ej2_instances[0]

/>treeObj.collapseAll("book"); |

| Prevent multiple nodes expand | Property: EnableMultipleExpand

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text")).EnableMultipleExpand(false) | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeExpanding("nodeExpanding").Render()

</Script>

function nodeExpanding(args){
var
parent=args.node.parentNode.closest('.e-list-item');
if (parent === null)
parent=args.node.parentNode.closest('.e-treeview');
var
children=parent.querySelectorAll('.e-list-item');
var i=0;
var nodes=[];

/>while(i<children.length){
nodes.push(children[i].getAttribute("data-uid"));
i++;}

/>this.collapseAll(nodes)
} |

| Expand all Nodes | Method: expandAll

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$(("#tree").data("ejTreeView"));
treeObj.expandAll(); | Method: expandAll

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var
treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.expandAll(); |

| Expand Node | Method: expandNode

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$(("#tree").data("ejTreeView"));
treeObj.expandNode("1"); | Method: expandAll

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>

var
treeObj = document.getElementById('tree').ej2_instances[0]
treeObj.expandAll("1"); |

| Gets/Sets Expanded nodes | Property: ExpandedNodes

List<int> ExpandedNodes = new
List<int>();
ExpandedNodes.Add(1);
ExpandedNodes.Add(2);

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text")).ExpandedNodes(ViewBag.ExpandedNodes) |
Property: ExpandedNodes

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ExpandedNodes("1").Render() |

| Expand action | Property: ExpandOn

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)

/>.Id("Id").ParentId("Parent").Text("Text")).ExpandOn("click") | Property: ExpandOn


```

```
/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).ExpandOn(ExpandOnSettings.Click).Render() |
```

```
| Get expanded nodes | Method: getExpandedNodes

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getExpandedNodes(); | Can be achieved using,

/>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var
instance= document.getElementById("tree").ej2_instances[0];
var
expand=instance.expandedNodes;
var i=0;
var expandedNodes;

/>while(i<expand.length) {
expandedNodes.push(instance.element.querySelector("[data-
uid="" + expandednodes[i] + ""']));
i++;
}
console.log(expandedNodes) |
```

```
| Get expanded nodes index | Method: getExpandedNodesIndex

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getExpandedNodesIndex(); | Can be achieved
using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var instance= document.getElementById("tree").ej2_instances[0];
var
nodes=instance.element.querySelectorAll('.e-list-item')
var nodeIndex;
var i = 0;

/>while(i<nodes.length) {
if(nodes[i].classList.contains('e-icon-collapsible'))
{nodeIndex.push(i);
}
i++;
}
console.log(nodeIndex) |
```

```
| To check if node is expanded | Method: isExpanded

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
var treeObj =
$("#tree").data("ejTreeView");
treeObj.isExpanded("book"); | Can be achieved using,

/>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
var
instance= document.getElementById("tree").ej2_instances[0];

/>if(instance.expandedNodes.indexOf("book") !== -1) {
console.log("Node is
expanded")
} |
```

```
| Triggers when node is collapsed successfully | Event: NodeCollapse

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
.<ClientSideEvents(events
=>events.NodeCollapse("nodeCollapse"))
function nodeCollapse() {} | Event:
NodeCollapsed

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeCollapsed("nodeCollapsed").Rende
r()

</Script>

function nodeCollapsed(){} |
```

```
| Triggers when node is expanded successfully | Event: NodeExpand

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

```

```

/>.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.NodeExpand("nodeExpand"))
function nodeExpand() {} | Event: NodeExpanded

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeExpanded("nodeExpanded").Render()

</Script>

function nodeExpanded(){} |

```

### Node Editing

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Editing | **Property:** AllowEditing<br /><br />@Html.EJ().TreeView("tree").TreeViewFields(field=><br />field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)<br />.Id("Id").ParentId("Parent").Text("Text")).AllowEditing(true) | **Property:** AllowEditing<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowEditing(true).Render() |

| Triggers before node is edited | **Event:** BeforeEdit<br /><br />@Html.EJ().TreeView("tree").TreeViewFields(field=><br />field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)<br />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events=>events.BeforeEdit("beforeEdit"))<br />function beforeEdit() {} | **Event:** NodeEditing<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeEditing("nodeEditing").Render()<br /><br /></Script><br /><br />function nodeEditing(){} |

| To Enable editing programatically | Not Applicable | **Method:** beginEdit<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()<br /><br /></Script><br /><br />var treeObj = document.getElementById('tree').ej2\_instances[0]<br />treeObj.beginEdit("1"); |

| Triggers before node edit is successful | **Event:** InlineEditValidation<br /><br />@Html.EJ().TreeView("tree").TreeViewFields(field=><br />field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)<br />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events=>events.InlineEditValidation("inlineEditValidation"))<br />function inlineEditValidation() {} | **Event:** NodeEdited<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeEdited("nodeEdited").Render()<br /><br /></Script><br /><br />function nodeEdited(){} |

| Triggers when node is edited successfully | **Event:** NodeEdit<br /><br />@Html.EJ().TreeView("tree").TreeViewFields(field=><br />field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)<br />.Id("Id").ParentId("Parent").Text("Text"))<br />.ClientSideEvents(events=>events.NodeEdit("nodeEdit"))<br />function nodeEdit() {} | **Event:** DataSourceChanged<br /><br />@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).DataSourceChanged("DataSourceChanged").Render()<br /><br /></Script><br /><br />function DataSourceChanged(args){} |

### Node Selection

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

```

| Multi-selection | Property: AllowMultiSelection

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).AllowMultiSelection(true) | Property:
AllowMultiSelection

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).Render() |

| Triggers before node is selected | Event: BeforeSelect

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events
=>events.BeforeSelect("beforeSelect"))
function beforeSelect() {} | Event: NodeSelecting

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeSelecting("nodeSelecting").Render()
()

Script

function nodeSelecting(){} |

| Fullrowselection | Property: FullRowSelect

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text")).FullRowSelect(true) | Property: FullRowSelect

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).FullRowSelect(true).Render() |

| Get selected node | Method: getSelectedNode

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getSelectedNode(); | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var
instance= document.getElementById("tree").ej2_instances[0];
var
select=instance.selectedNodes;
var selectedNode;
selectedNode.push(instance.element.querySelector('[data-uid="'+ selectedNode[i] +
"']));
console.log(selectedNode) |

| Get selected node index | Method: getSelectedNodeIndex

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)
.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.getSelectedNodeIndex(); | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var
instance= document.getElementById("tree").ej2_instances[0];
var
nodes=instance.element.querySelectorAll('.e-list-item')
var nodeIndex;
var i=0;
while(i<nodes.length) {
if(nodes[i].classList.contains('e-active')) {nodeIndex = i;
break;
}
i++
}
console.log(nodeIndex) |

| Get selected nodes | Method: getSelectedNodes

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br

```

```

/>.Id("Id").ParentId("Parent").Text("Text")).AllowMultiSelection(true))

</Script>
</var>
treeObj = $("#tree").data("ejTreeView");
</treeObj.getSelectedNodes(); | Can be achieved
using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).Render()

</>
</Script>
</var> instance= document.getElementById("tree").ej2_instances[0];
</var>
select=instance.selectedNodes;
</var> i=0;
</var> selectedNodes;

/>while(i<select.length) {
</selectedNodes.push(instance.element.querySelector('[data-
uid="'+ selectedNodes[i] + "')]);
</i++
</></br /></console.log(selectedNodes) |

| Get selected nodes index | Method: getSelectedNodesIndex

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).AllowMultiSelection(true))

</Script>
</var>
treeObj = $("#tree").data("ejTreeView");
</treeObj.getSelectedNodesIndex(); | Can be
achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).Render()

</>
</Script>
</var> instance= document.getElementById("tree").ej2_instances[0];
</var>
nodes=instance.element.querySelectorAll('.e-list-item')
</var> nodeIndex;
</var> i = 0;

/>while(i<nodes.length) {
</if(nodes[i].classList.contains('e-active')) {nodeIndex.push(i);

/></br /></i++
</></br /></console.log(nodeIndex) |

| To check if node is selected | Method: isSelected

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))

</Script>
</var> treeObj =
$("#tree").data("ejTreeView");
</treeObj.isSelected("book"); | Can be achieved using,

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

</Script>
</var>
instance= document.getElementById("tree").ej2_instances[0];
</if>
(instance.selectedNodes.indexOf("book") !== -1) {
</console.log("Node is selected")
</> |

| Triggers when node is selected successfully | Event: NodeSelect

/>@Html.EJ().TreeView("tree").TreeViewFields(field =>

/>field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text"))
</>.ClientSideEvents(events
=>events.NodeSelect("nodeSelect"))
</function nodeSelect() {} | Event: NodeSelected

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeSelected("nodeSelected").Render(
)

</Script>

</function nodeSelected(){
</if (args.action == "select") {}
</> |

| Select all nodes | Method: selectAll

/>@Html.EJ().TreeView("tree").TreeViewFields(field
=>
</field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)

/>.Id("Id").ParentId("Parent").Text("Text")).AllowMultiSelection(true))

</Script>
</var>
treeObj = $("#tree").data("ejTreeView");
</treeObj.selectAll(); | Can be achieved using,

</>

/>@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).Render()

</>
</Script>
</var> instance= document.getElementById("tree").ej2_instances[0];
</var>
nodes = instance.element.querySelectorAll('.e-list-item')
</var> selectednodes;
</for(int i =

```



```
0; i < nodes.length; i++) {
selectednodes.push(nodes[i].getAttribute('data-uid'))}
instance.selectedNodes = selectednodes; |
```

```
| Gets/Sets selected node | Property: SelectedNode

List<int> SelectedNode = new
List<int>();
SelectedNode.Add(1);
@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br

.Id("Id").ParentId("Parent").Text("Text")).SelectedNode(ViewBag.SelectedNode) | Property:
SelectedNodes

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).SelectedNodes("1").Render() |
```

```
| Select node | Method: selectNode

@Html.EJ().TreeView("tree").TreeViewFields(field
=>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br
</>.Id("Id").ParentId("Parent").Text("Text"))

Script
var treeObj =
$("#tree").data("ejTreeView");
treeObj.selectNode("book"); | Can be achieved using,<br
</>
@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).Render()

Script
var
instance= document.getElementById("tree").ej2_instances[0];<br
</>instance.selectedNodes=["book"] |
```

```
| Gets/Sets selected nodes | Property: SelectedNodes

List<int> SelectedNode = new
List<int>();
SelectedNode.Add(1);
SelectedNode.Add(2);<br
</@Html.EJ().TreeView("tree").TreeViewFields(field =><br
</field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)<br
</>.Id("Id").ParentId("Parent").Text("Text")).SelectedNodes(ViewBag.SelectedNode)) | Property:
SelectedNodes
<br
</@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).SelectedNode
s(["1","2"]).Render(); |
```

```
| Triggers when node is unselected successfully | Event: NodeUnselect

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text"))
.ClientSideEvents(events =>events.NodeUnselect("nodeUnselect"))
function nodeUnselect() {} | Event: NodeSelected

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeSelected("nodeSelected").Render(

Script

function nodeSelected(){
if (args.action == "un-select") {}
}
```

```
| To unselect all nodes | Method: unselectAll

@Html.EJ().TreeView("tree").TreeViewFields(field =>
field.Datasource((IEnumerable<LoadData>)ViewBag.datasource)
.Id("Id").ParentId("Parent").Text("Text")).AllowMultiSelection(true)

Script
var
treeObj = $("#tree").data("ejTreeView");
treeObj.unselectAll(); | Can be achieved using,

@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).Render()

Script
var instance= document.getElementById("tree").ej2_instances[0];
instance.selectedNodes=[]; |
```

| To unselect node | **Method:** `unselectNode`  
`</@Html.EJ().TreeView("tree").TreeViewFields(field =>  
 </field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 </>.Id("Id").ParentId("Parent").Text("Text")).AllowMultiSelection(true))<br /></Script><br />var  
 treeObj = $("#tree").data("ejTreeView");<br />treeObj.unselectNode("book"); | Can be achieved  
 using,<br /><br />  
</@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).AllowMultiSelection(true).Render()<br />  
 </></Script><br />var instance= document.getElementById("tree").ej2_instances[0];<br />  
 </instance.selectedNodes.pop(book) |`

## Template

| Behavior | API in Essential JS 1 | API in Essential JS 2 |

| --- | --- | --- |

| Custom template | **Property:** `Template`  
`</@Html.EJ().TreeView("tree").TreeViewFields(field =>  
 </field.Datasource((IEnumerable<LoadData>)ViewBag.datasources)<br />  
 </>.Id("Id").ParentId("Parent").Text("Text")).Template("templateData") | Property:  
NodeTemplate<br /></>  
 </@Html.EJS().TreeView("tree").Fields(ViewBag.Fields).NodeTemplate("templateData").Render  
 () |`

## Uploader

### Getting Started with ASP.NET MVC Uploader Control

This section briefly explains about how to include [ASP.NET MVC Uploader](#) control in your ASP.NET MVC application using Visual Studio.

#### Prerequisites

[System requirements for ASP.NET MVC controls](#)

Create ASP.NET MVC application with HTML helper

- [Create a Project using Microsoft Templates](#)
- [Create a Project using Syncfusion ASP.NET MVC Extension](#)

#### Install ASP.NET MVC package in the application

To add ASP.NET MVC controls in the application, open the NuGet package manager in Visual Studio (Tools → NuGet Package Manager → Manage NuGet Packages for Solution), search for [Syncfusion.EJ2.MVC5](#) and then install it. Alternatively, you can utilize the following package manager command to achieve the same.

#### PACKAGE MANAGER

```
Install-Package Syncfusion.EJ2.MVC5 -Version {{ site.ej2version }}
```

**Note:** Syncfusion ASP.NET MVC controls are available in [nuget.org](#). Refer to [NuGet packages topic](#) to learn more about installing NuGet packages in various OS environments. The Syncfusion.EJ2.MVC5

NuGet package has dependencies, [Newtonsoft.Json](#) for JSON serialization and [Syncfusion.Licensing](#) for validating Syncfusion license key.

**Note:** If you create ASP.NET MVC application with MVC4 package, search for [Syncfusion.EJ2.MVC4](#) and then install it.

#### Add namespace

Add **Syncfusion.EJ2** namespace reference in **Web.config** under **Views** folder.

`

```
<namespaces>
```

```
<add namespace="Syncfusion.EJ2"/>
```

```
</namespaces>
```

`

#### Add stylesheet and script resources

Here, the theme and script is referred using CDN inside the **<head>** of **~/Pages/Shared/\_Layout.cshtml** file as follows,

##### ~/ LAYOUT.CSHTML

```
<head>
...
<!-- Syncfusion ASP.NET MVC controls styles -->
<link rel="stylesheet" href="https://cdn.syncfusion.com/ej2/{{
site.ej2version }}/fluent.css" />
<!-- Syncfusion ASP.NET MVC controls scripts -->
<script src="https://cdn.syncfusion.com/ej2/{{ site.ej2version
}}/dist/ej2.min.js"></script>
</head>
```

**Note:** Checkout the [Themes topic](#) to learn different ways (CDN, NPM package, and [CRG](#)) to refer styles in ASP.NET MVC application, and to have the expected appearance for Syncfusion ASP.NET MVC controls. Checkout the [Adding Script Reference](#) topic to learn different approaches for adding script references in your ASP.NET MVC application.

#### Register Syncfusion script manager

Also, register the script manager **EJS().ScriptManager()** at the end of **<body>** in the **~/Pages/Shared/\_Layout.cshtml** file as follows.

##### ~/ LAYOUT.CSHTML

```
<body>
...
<!-- Syncfusion ASP.NET MVC Script Manager -->
@Html.EJS().ScriptManager()
</body>
```

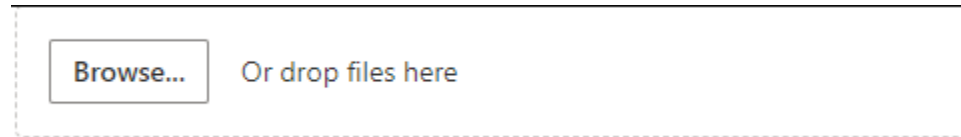
#### Add ASP.NET MVC Uploader control

Now, add the Syncfusion ASP.NET MVC Uploader control in **~/Views/Home/Index.cshtml** page.

**CSHTML**

```
@Html.EJS().Uploader("UploadFiles").Render()
```

Press Ctrl+F5 (Windows) or ⌘+F5 (macOS) to run the app. Then, the Syncfusion ASP.NET MVC Uploader control will be rendered in the default web browser.

**Adding drop area**

By default, the uploader control allows to upload files by drag the files from file explorer, and drop into the drop area. You can configure any other external element as drop target using [DropArea](#) property.

**CSHTML**

```
<div id='droparea'>Drop files here to upload</div>
@Html.EJS().Uploader("UploadFiles").DropArea("#droparea").AutoUpload(false).
Render()
<style>
 .fileupload {
 margin: 20px auto;
 width: 400px;
 }
 #droparea {
 padding: 50px 25px;
 margin: 30px auto;
 border: 1px solid #c3c3c3;
 text-align: center;
 width: 20%;
 display: inline-flex;
 }
 .e-file-select,
 .e-file-drop {
 display: none;
 }
 body .e-upload-drag-hover {
 outline: 2px dashed brown;
 }
 #uploadfile {
 width: 60%;
 display: inline-flex;
 margin-left: 5%;
 }
</style>
```

**Configure asynchronous settings**

The uploader control process the files to upload in Asynchronous mode by default. Define the properties [SaveUrl](#) and [RemoveUrl](#) to handle the save and remove action as follows.

**CSHTML**

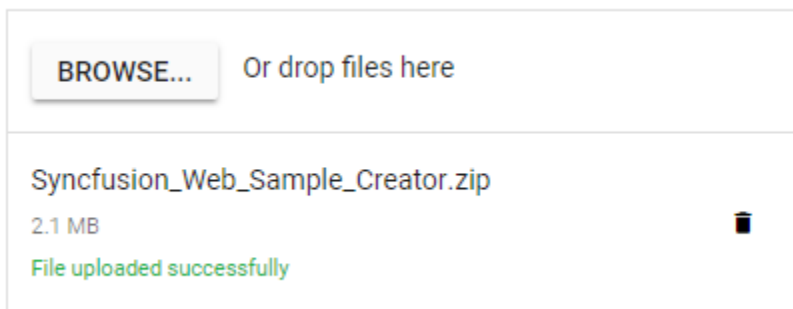
```
@Html.EJS().Uploader("UploadFiles").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove"
}).AutoUpload(false).Render()
```

### Handle success and failed upload

You can handle the success and failure actions using the [Success](#) and [Failure](#) events. To handle these events, define the function and assign it to the corresponding event as follows.

#### CSHTML

```
@Html.EJS().Uploader("UploadFiles").Success("onUploadSuccess").Failure("onUp
loadFailed").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
<script>
 function onUploadSuccess(args) {
 if (args.operation === 'upload') {
 console.log('Success');
 }
 }
 function onUploadFailed(args) {
 console.log('Failed');
 }
</script>
```



**Note:** [View Sample in GitHub](#).

See also

- [How to add additional data on upload](#)
- [Achieve file upload programmatically](#)
- [Achieve invisible upload](#)

You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) that shows how to render the file upload and browse the files which you want to upload to the server.

### Asynchronous upload

The uploader control allows you to upload the files asynchronously.

The upload process requires save and remove action URL to manage the upload process in the server.

- The save action is necessary to handle the upload operation.
- The remove action is optional, one can handle the removed files from server.

The File can be uploaded automatically or manually. For more information, you can refer to the [Auto Upload](#) section from the documentation.

### Multiple file upload

By Default, the uploader control allows you to select and upload multiple files simultaneously.

The selected files are organized in a list for every file selection until you clear it by clicking clear button that is shown in footer. You can add the multiple attributes to original input element of file by enabling the multiple file selection.

The following example explains about [multiple](#) file upload settings.

#### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

#### MULTIPLE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

### Single file upload

You can select and upload a single file by disabling the [multiple](#) file selection property.

The file list item is removed for every selection and it always maintain a single file to upload.

You can remove the multiple attributes form the original input element of file by enabling the single file upload property.

The following example explains about single file upload settings.

**CSHTML**

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).Multiple(false).AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

**SINGLE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

**Save action**

The save action handler upload the files that needs to be specified in the [saveUrl](#) property.

The save handler receives the submitted files and manages the save process in server.

After uploading the files to server location, the color of the selected file name changes to green and the remove icon is changed as bin icon.

- When the file is uploaded successfully, the event [success](#) triggers to handle the operation after upload.
- When the file is failed to upload, the event [failure](#) triggers with information, which cause this failure.

You can cancel the upload process by setting the upload event argument [eventargs.cancel](#) to true.

**CSHTML**

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Save" }).Render()
```

**SAVE.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
```

```
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

#### *Server-side configuration for save action*

This section explains how to handle the server-side action for saving the file from server.

```
`csharp
[AcceptVerbs("Post")]
public void Save()
{
 try
 {
 var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
 if (httpPostedFile != null)
 {
 var fileSave = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");
 var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
 if (!System.IO.File.Exists(fileSavePath))
 {
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File uploaded succesfully";
 Response.End();
 }
 else
 {
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "204 File already exists";
 Response.StatusCode = 204;
 }
 }
 }
}
```



```

Response.StatusDescription = "File already exists";
Response.End();
}
}
}
catch (Exception e)
{
 HttpResponse Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 204;
 Response.Status = "204 No Content";
 Response.StatusDescription = e.Message;
 Response.End();
}
}
,

```

### Remove action

The remove action is optional. Specify the URL to handle remove process from server.

The remove handler receives the posted files and handle the remove operation in server.

- When the files are removed successfully from the server, the [success](#) event triggers to denote the process has completed.
- When remove action fails, the event [failure](#) triggers with information, which cause failure in remove process.

**Note:** You can differentiate the file operation whether the success event triggers from save or remove action in its arguments [eventArgs.operation](#).

You can remove the files which is not uploaded locally by clicking the remove icon. In this case, the success or failure events will not be triggered.

### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

### REMOVE.CS

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

#### *Server-side configuration for remove action*

This section explains how to handle the server-side action for removing the file from server.

```
`csharp
[AcceptVerbs("Post")]
public void Remove()
{
 try
 {
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "200 OK";
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File removed succesfully";
 Response.End();
 }
 catch (Exception e)
 {
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "200 OK";
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File removed succesfully";
 }
}
```

```
Response.End();
}
}
,
```

### Auto upload

By default, the uploader processes the files to upload once the files are selected and added in upload queue. To upload manually, disable the [autoUpload](#) property. When you disable this property, you can use the action buttons to call upload all or clear all actions manually. You can change those buttons text using the [buttons](#) property in Uploader control.

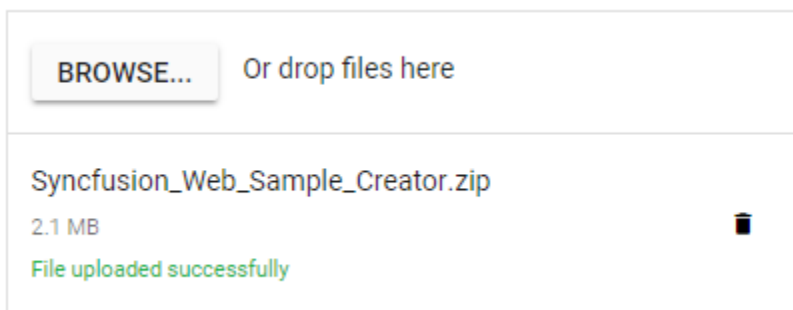
### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).Render()
```

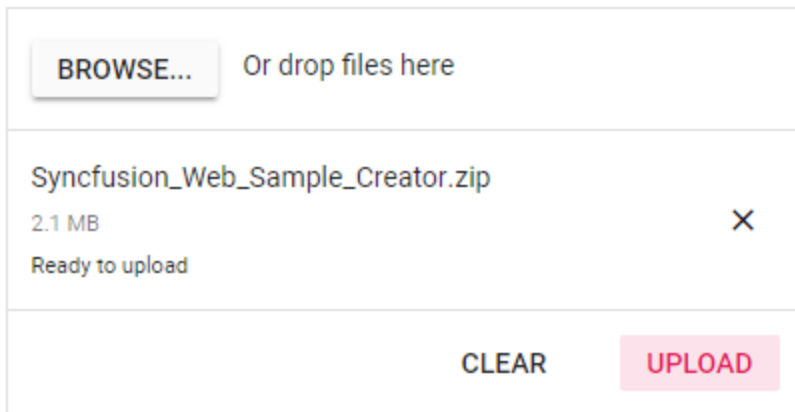
### AUTO-UPLOAD.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Auto upload output be like the below.



Auto upload false output be like the below.



### Sequential upload

By default, the uploader control process multiple files to upload simultaneously. When you enable the [sequentialUpload](#) property, the selected files will process sequentially (one after the other) to the server. If the file uploaded successfully or failed, the next file will upload automatically in this sequential upload. This feature helps to reduce the upload traffic and reduce the failure of file upload.

### CSHTML

```
@Html.EJS().Uploader("UploadFiles").SequentialUpload(true).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove"}).AutoUpload(false).Render()
```

### SEQUENTIAL-UPLOAD.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult SequentialUpload()
 {
 return View();
 }
 }
}
```

### Preload files

The uploader control allows you to preload the list of files that are uploaded in the server. The preloaded files are useful to view and remove the files from server that can be achieved by the [files](#) property. By default, the files are configured with uploaded successfully state on rendering file list. The following properties are mandatory to configure the preloaded files:

- Name
- Size
- Type

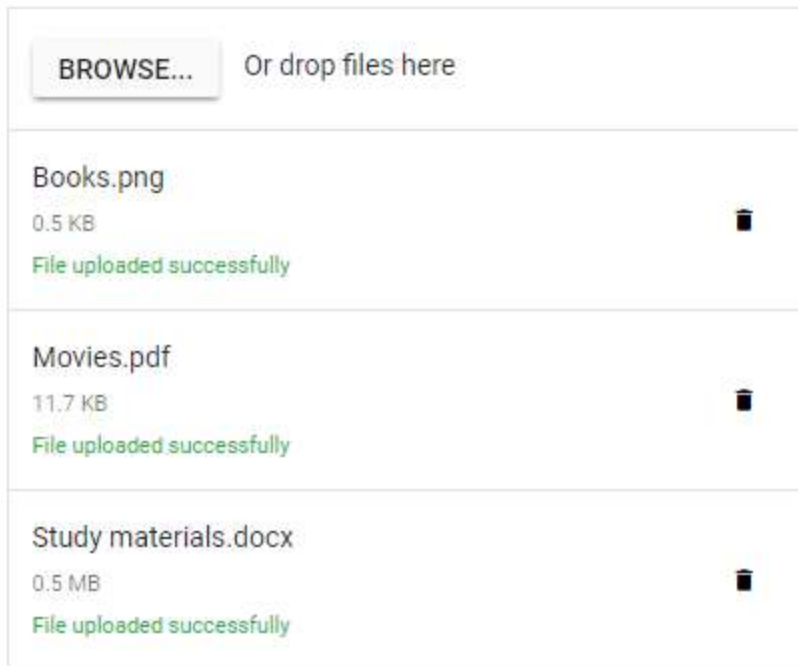
### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove"
}).Files(@ViewBag.datasource).Render()
```

### PRELOAD-FILE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 List<UploaderUploadedFiles> list = new
List<UploaderUploadedFiles>();
 public ActionResult DefaultFunctionalities()
 {
 list.Add(new UploaderUploadedFiles { Name = "Books", Size =
500000, Type = ".png" });
 list.Add(new UploaderUploadedFiles { Name = "Movies", Size =
12000, Type = ".pdf" });
 list.Add(new UploaderUploadedFiles { Name = "Study materials",
Size = 500000, Type = ".docx" });
 ViewBag.datasource = list;
 return View();
 }
 }
}
```

Output be like the below.



### Adding additional HTTP headers with upload action

The Uploader control allows you to add the additional headers with **save** and **remove** action requests using the [uploading](#) and [removing](#) events, which helps to send validation token on file upload. Access the current request and set the request header within these events.

The following code block shows how to add the additional headers with save and remove action request.

```
`html
@Html.EJS().Uploader("UploadFiles").DropArea(".control-
fluid").Uploading("addHeaders").Removing("addHeaders").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
,

`javascript
function addHeaders(args) {
args.currentRequest.setRequestHeader('custom-header', 'Syncfusion');
}
,
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

## See Also

- [How to add additional data on upload](#)
- [How to add confirm dialog to remove the files](#)
- [Check the MIME type of file before uploading it](#)
- [How to open and edit the uploaded files](#)

## Chunk Upload

The Uploader sends the large file split into small chunks and transmits to the server using AJAX. You can also pause, resume, and retry the failed chunk file.

**Note:** \* The chunk upload works in asynchronous upload only.

- This feature is available from the Essential Studio Vol 2, 2018 release.

To enable the chunk upload, set the size to [chunkSize](#) option of the upload and it receives the value in bytes.

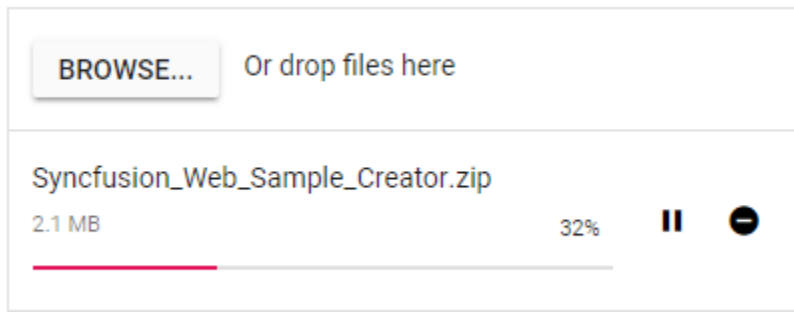
**CSHTML**

```
@Html.EJS().Uploader("UploadFiles").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove", ChunkSize =
102400 }).AutoUpload(false).Render()
```

**CHUNK.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



The chunk upload functionality separates the selected files into blobs of the data or chunks. These chunks are transmitted to the server using an AJAX request.

The chunks are sent in **sequential** order, and the next chunk can be sent to the server according to the [success](#) of the previous chunk. If any one of the chunk failed, then the remaining chunk cannot be sent to the server.

The [chunkSuccess](#) or [chunkFailure](#) event will be triggered when the chunk is sent to the server successfully or failed. If all the chunks are sent to the server successfully, the uploader success event is triggered.

**Note:** Chunk upload will work when the selected file size is greater than the specified chunk size. otherwise, it upload the files normally.

#### Additional configurations

To modify the chunk upload, the following options can be used.

- **RetryAfterDelay** - If error occurs while sending any chunk request from JavaScript, hold the operation for 500 milliseconds (by default), and retry the operation using chunk. This can be achieved by using the [asyncSettings.retryAfterDelay](#) property. You can modify the holding time interval in milliseconds.
- **RetryCount** - Specifies the number of retry actions performed when the file fails to upload. By default, retry action is performed 3 times. If the file fails to upload continuously, the request is aborted and the uploader [failure](#) event will trigger.

The following sample specifies the chunk upload delay with 3000 milliseconds and the retry count is 5. The failure event is triggered as the wrong saveUrl is used.

#### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove", ChunkSize =
102400, RetryCount = 5, RetryAfterDelay = 3000 }).AutoUpload(false).Render()
```

#### RETRY.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
```



```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

### Resumable upload

Allows you to resume an upload operation after a network failure or manually interrupts (pause) the upload. You can perform pause and resume upload actions using public methods (pause and resume) and UI interaction. The pause icon is enabled after the upload begins.

**Note:** This pause and resume features available only when the chunk upload is enabled.

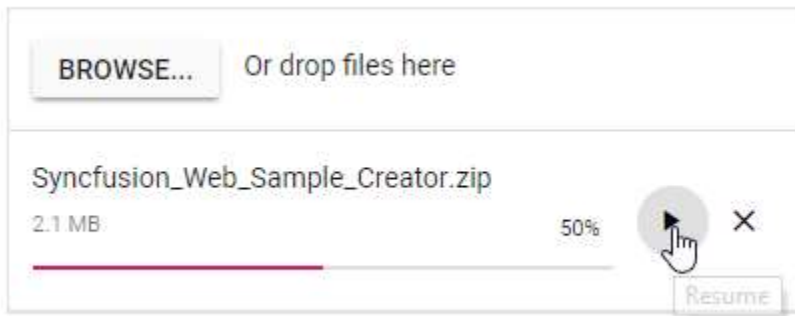
### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove", ChunkSize =
102400 }).AutoUpload(false).Render()
```

### RESUMABLE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



### Cancel upload

The uploader component allows you to cancel the uploading file. This can be achieved by clicking the cancel icon or using the `cancel` method. The [cancelling](#) event will be fired whenever the file upload request is canceled. While canceling the upload request, the partially uploaded file is removed from the server.

When the request fails, the pause icon is changed to retry icon. By clicking the retry icon, sends the failed chunk request again to the server and upload started from where it is failed. You can retry the canceled upload request again using retry UI or `retry` methods. But, if you retry this, the file upload action again starts from initial.

The following example explains about chunk upload with cancel support.

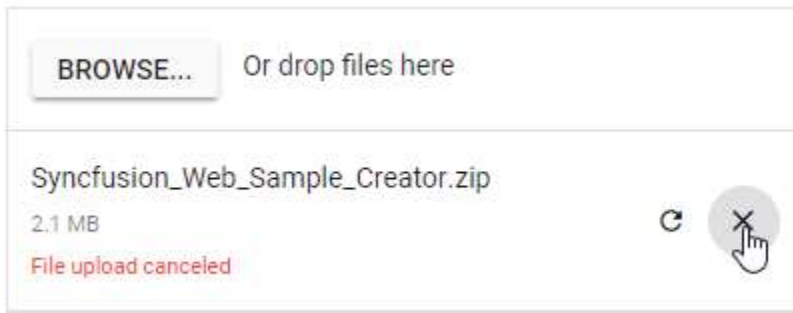
### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove", ChunkSize =
102400 }).AutoUpload(false).Render()
```

### CANCEL.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



**Note:** The retry action has different working behavior for chunk upload and default upload.

- Chunk upload - Retries to upload the failed request where it is failed previously.
- Default upload - Retries to upload the failed file again from initial.

### Server-Side configurations

The server-side implementation entirely depends on the application requirements and logic. The following code snippet provides the server-side logic to handle the chunk upload using the uploader components.

```
`csharp
// Server configuration for upload a file.
public void Save()
{
 try
 {
 if (System.Web.HttpContext.Current.Request.Files.AllKeys.Length > 0)
 {
 var httpPostedChunkFile = System.Web.HttpContext.Current.Request.Files["chunkFile"];
 if (httpPostedChunkFile != null)
 {
 var saveFile = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");
 var SaveFilePath = Path.Combine(saveFile, httpPostedChunkFile.FileName + ".part");
 var chunkIndex = System.Web.HttpContext.Current.Request.Form["chunk-index"];
 if (chunkIndex == "0")
 {
 //httpPostedChunkFile.SaveAs(SaveFilePath);
 }
 }
 else
 {
 }
 }
 }
}
```

```
{
// MergeChunkFile(SaveFilePath, httpPostedChunkFile.InputStream);
var totalChunk = System.Web.HttpContext.Current.Request.Form["total-chunk"];
if (Convert.ToInt32(chunkIndex) == (Convert.ToInt32(totalChunk) - 1))
{
var savedFile = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");
var originalFilePath = Path.Combine(savedFile, httpPostedChunkFile.FileName);
System.IO.File.Move(SaveFilePath, originalFilePath);
}
}

HttpResponse ChunkResponse = System.Web.HttpContext.Current.Response;
ChunkResponse.Clear();
ChunkResponse.ContentType = "application/json; charset=utf-8";
ChunkResponse.StatusDescription = "File uploaded succesfully";
ChunkResponse.End();
}

var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];
if (httpPostedFile != null)
{
var fileSave = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
if (!System.IO.File.Exists(fileSavePath))
{
// httpPostedFile.SaveAs(fileSavePath);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
Response.StatusDescription = "File uploaded succesfully";
Response.End();
}
else
{
HttpResponse Response = System.Web.HttpContext.Current.Response;
```

```
Response.Clear();
Response.Status = "204 File already exists";
Response.StatusCode = 204;
Response.StatusDescription = "File already exists";
Response.End();
}
}
}
}
catch (Exception e)
{
 HttpResponse Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusCode = 204;
 Response.Status = "204 No Content";
 Response.StatusDescription = e.Message;
 Response.End();
}
}

// Server configuration for remove a uploaded file
public void Remove()
{
 try
 {
 HttpResponse Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "200 OK";
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File removed succesfully";
 Response.End();
 }
}
```

```
catch (Exception e)
{
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.Status = "200 OK";
 Response.StatusCode = 200;
 Response.ContentType = "application/json; charset=utf-8";
 Response.StatusDescription = "File removed succesfully";
 Response.End();
}
}

// Merge the current chunk file with previous uploaded chunk files
public void MergeChunkFile(string fullPath, Stream chunkContent)
{
 try
 {
 using (FileStream stream = new FileStream(fullPath, FileMode.Append, FileAccess.Write,
 FileShare.ReadWrite))
 {
 using (chunkContent)
 {
 chunkContent.CopyTo(stream);
 }
 }
 }
 catch (IOException ex)
 {
 throw ex;
 }
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

## File source

### Paste to upload

The uploader control allows you to upload the files using the select or drop files option from the file explorer. It also supports pasting to upload the image files. You can upload any currently copied images in the clipboard.

**Note:** When you paste the image, it will be saved in the server with the filename as `image.png`. The file name can be renamed in the server end. You can generate a random name for the file name using `getUniqueID` method.

Refer to the following example.

### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).Uploading("onUploadBegin").DirectoryUpload(true).AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
<script>
 function onUploadBegin(args) {
 // check whether the file is uploading from paste.
 if (args.fileData.fileSource === 'paste') {
 let newName = getUniqueID(args.fileData.name.substring(0, args.fileData.name.lastIndexOf('.'))) + '.png';
 args.customFormData = [{ 'fileName': newName }];
 }
 }
</script>
```

### PASTE.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Uploader
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

*Server-side configuration for save the paste file*

``csharp`

`[AcceptVerbs("Post")]`

`public void Save()`

```

{
var httpPostedFile = HttpContext.Current.Request.Files["UploadFiles"];
var fileSave = HttpContext.Current.Server.MapPath("UploadedFiles");
var fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);
if (!System.IO.File.Exists(fileSavePath))
{
httpPostedFile.SaveAs(fileSavePath);
// Get the current file name
var oldName = httpPostedFile.FileName;
// Get the additional data as name in server end by corresponding key.
var newName = HttpContext.Current.Request.Form["fileName"];
// Rename the file
File.Move(oldName, newName);
HttpResponse Response = System.Web.HttpContext.Current.Response;
Response.Clear();
Response.ContentType = "application/json; charset=utf-8";
// Sending the file path to client side
Response.StatusDescription = fileSavePath;
Response.End();
}
}
`

```

### Directory upload

The uploader control allows you to upload all files in the folders to server by using the [directoryUpload](#) property. When this property is enabled, the uploader control processes the files by iterating through the files and sub-directories in a directory. It allows you to select only folders instead of files to upload.

**Note:** The directory upload is available only in browsers that supports **HTML5 directory**. The uploader will process directory upload by dragging and dropping in the Edge browser. Refer to the following example to upload files to the server.

### CSHTML

```

@Html.EJS().Uploader("UploadFiles").AutoUpload(false).DirectoryUpload(true).
AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()

```

### DIRECTORY.CS



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.Uploader
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

*Server-side configuration for save the files of folders*

`csharp

[AcceptVerbs("Post")]

public void Save()

{

var httpPostedFile = System.Web.HttpContext.Current.Request.Files["UploadFiles"];

var fileSave = System.Web.HttpContext.Current.Server.MapPath("UploadedFiles");

// split the folders by using file name

string[] folders = httpPostedFile.FileName.Split('/');

string fileSavePath = "";

if (folders.Length > 1)

{

for (var i = 0; i < folders.Length - 1; i++)

{

var newFolder = Path.Combine(fileSave, folders[i]);

// create folder

Directory.CreateDirectory(newFolder);

fileSave = newFolder;

}

fileSavePath = Path.Combine(fileSave, folders[folders.Length - 1]);

}

else

{

fileSavePath = Path.Combine(fileSave, httpPostedFile.FileName);

```

}
if (!System.IO.File.Exists(fileSavePath))
{
 // save file in the corresponding server location
 httpPostedFile.SaveAs(fileSavePath);
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 // Sending the file path to client side
 Response.StatusDescription = fileSavePath;
 Response.End();
}
}
,

```

### Drag and drop

The uploader control allows you to drag and drop the files to upload.

You can drag the files from file explorer and drop into the drop area.

By default, the uploader control act as drop area element. The drop area gets highlighted when you drag the files over drop area.

### Custom drop area

The uploader control allows you to set external target element as drop area using the [dropArea](#) property. The element can be represented as HTML element or element's id.

### CSHTML

```

<div id="droparea">
 Drop files here to upload
</div>
@Html.EJS().Uploader("UploadFiles").DropArea("#droparea").AutoUpload(false).
Render()

```

### INDEX.CSS

```

#droparea {
 padding: 50px 25px;
 margin: 30px auto;
 border: 1px solid #c3c3c3;
 text-align: center;
 width: 20%;
 display: inline-flex;
}
.e-file-select,
.e-file-drop {
 display: none;
}

```

```

}
body .e-upload-drag-hover {
 outline: 2px dashed brown;
}
#uploadfile {
 width: 60%;
 display: inline-flex;
 margin-left: 5%;
}

```

Output be like the below.



### Customize drop area

You can customize the appearance of drop area by overriding the default drop area styles.

The class "" and "" is available to handle this customization.

### CSHTML

```

<div id='dropArea' style='height: auto; overflow: auto'>
 Drop files here or <a href=''
id='browse'><u>Browse</u>

@Html.EJS().Uploader("UploadFiles").AllowedExtensions(".jpg,.png").AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
 function browseClick() {
 document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click(); return false;
 }
</script>
<style>
 .e-file-select-wrap {
 display: none;
 }
 #dropArea .e-upload {
 border: 0;
 margin-top: 15px;
 }
 #drop {
 padding-left: 30%;
 }

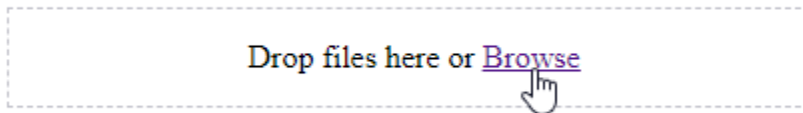
```

```
#dropArea {
 min-height: 18px;
 border: 1px dashed #c3c3cc;
 padding-top: 15px;
 margin: 20px auto;
 width: 400px;
}
</style>
```

### CUSTOMIZE-DROP.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

See Also

- [Achieve file upload programmatically](#)
- [Validate image/\\* on drop](#)

### Drag and drop in Uploader Control

The uploader component allows you to drag and drop the files to upload. You can drag the files from file explorer and drop into the drop area.

By default, the uploader component act as drop area element. The drop area gets highlighted when you drag the files over drop area.

### Custom drop area

The uploader component allows you to set external target element as drop area using the [dropArea](#) property. The element can be represented as HTML element or element's id.

#### CSHTML

```
<div id="droparea">
 Drop files here to upload
</div>
@Html.EJS().Uploader("UploadFiles").DropArea("#droparea").AutoUpload(false).
Render()
```

#### INDEX.CSS

```
#droparea {
 padding: 50px 25px;
 margin: 30px auto;
 border: 1px solid #c3c3c3;
 text-align: center;
 width: 20%;
 display: inline-flex;
}
.e-file-select,
.e-file-drop {
 display: none;
}
body .e-upload-drag-hover {
 outline: 2px dashed brown;
}
#uploadfile {
 width: 60%;
 display: inline-flex;
 margin-left: 5%;
}
```

### Customize drop area

You can customize the appearance of drop area by overriding the default drop area styles. The class `""` and `""` is available to handle this customization.

#### CSHTML

```
<div id='dropArea' style='height: auto; overflow: auto'>
 Drop files here or <a href=''
id='browse'><u>Browse</u>

@Html.EJS().Uploader("UploadFiles").AllowedExtensions(".jpg,.png").AsyncSettings(
new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
 function browseClick() {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click(); return false;
 }
}
```

```

</script>
<style>
 .e-file-select-wrap {
 display: none;
 }
 #dropArea .e-upload {
 border: 0;
 margin-top: 15px;
 }
 #drop {
 padding-left: 30%;
 }
 #dropArea {
 min-height: 18px;
 border: 1px dashed #c3c3cc;
 padding-top: 15px;
 margin: 20px auto;
 width: 400px;
 }
</style>

```

### **CUSTOMIZE-DROP.CS**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}

```

### Validation in Uploader Control

The uploader control validate the selected files size and extension using the [allowedExtensions](#), [minFileSize](#) and [maxFileSize](#) properties. The files can be validated before uploading to the server and can be ignored on uploading. Also, you can validate the files by setting the HTML attributes to the original input element. The validation process occurs on drag-and-drop the files also.

#### File type

You can allow the specific files alone to Upload using the [allowedExtensions](#) property. The extension can be represented as collection by comma separators. The uploader control filters the selected or dropped files to match against the specified file types and processes the upload operation. The validation happens when you specify value to inline attribute to accept the original input element.

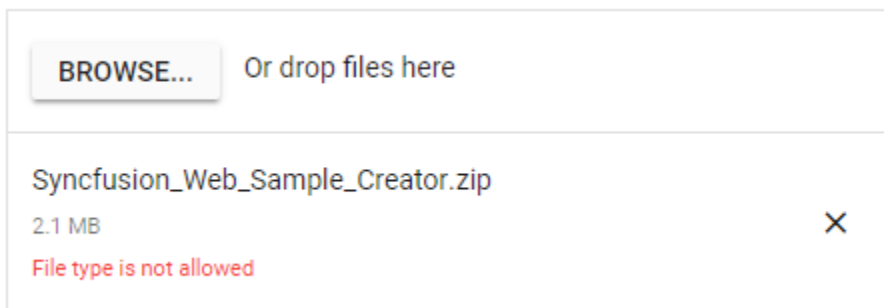
### **CSHTML**

```
@Html.EJS().Uploader("UploadFiles").AllowedExtensions(".doc, .docx, .xls,
.xlsx").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

### TYPE-VALIDATION.CS

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



### File size

The uploader control allows you to validate the files based on its size. The validation helps to restrict uploading large files or empty files to the server. The size can be represented in **bytes**. By default, the uploader control allows you to upload **minimum file size** as 0 byte and **maximum file size** as 28.4 MB using the [minFileSize](#) and [maxFileSize](#) properties.

### CSHTML

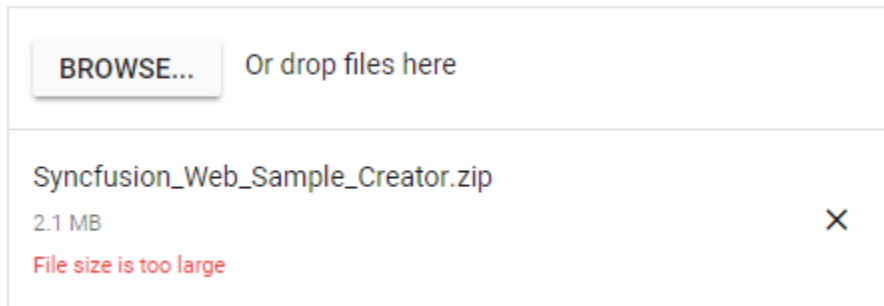
```
@Html.EJS().Uploader("UploadFiles").MinFileSize(10000).MaxFileSize(1000000).
AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

### SIZE-VALIDATION.CS

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

Output be like the below.



### Maximum files count

You can restrict uploading the maximum number of files using the **selected** event. In the selected event arguments, you can get the currently selected files details using the `getFilesData()`. You can modify the files details and assign the modified file list to the `eventArgs.modifiedFilesData`.

### CSHTML

```
@Html.EJS().Uploader("UploadFiles").Selected("onFileSelected").Selected("onFileSelected").Success("onUploadSuccess").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
<script>
 function onFileSelected(args) {
 // Filter the 5 files only to showcase
 var uploadObj = document.getElementById("UploadFiles");
 args.filesData.splice(5);
 var filesData = uploadObj.ej2_instances[0].getFilesData();
 var allFiles = filesData.concat(args.filesData);
 if (allFiles.length > 5) {
 for (var i = 0; i < allFiles.length; i++) {
 if (allFiles.length > 5) {
 allFiles.shift();
 }
 }
 }
 args.filesData = allFiles;
 }
}
```



```

 // set the modified custom data
 args.modifiedFilesData = args.filesData;
 }
 args.isModified = true;
}
function onUploadSuccess(args) {
 var _this = this;
 var li = this.uploadWrapper.querySelector('[data-file-name="' +
args.file.name + '"]');
 if (args.operation === 'upload') {
 li.querySelector('.e-file-delete-btn').onclick = function () {
 generateSpinner(_this.uploadWrapper);
 };
 li.querySelector('.e-file-delete-btn').onkeydown = function (e)
{
 if (e.keyCode === 13) {
 generateSpinner(e.target.closest('.e-upload'));
 }
 };
 }
 else {
 ej.popups.hideSpinner(this.uploadWrapper);
 ej.base.detach(this.uploadWrapper.querySelector('.e-spinner-
pane'));
 }
}
function generateSpinner(targetElement) {
 ej.popups.createSpinner({ target: targetElement, width: '25px' });
 ej.popups.showSpinner(targetElement);
}
</script>

```

### MAX-COUNT-VALIDATION.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}

```

### Duplicate files

You can validate the duplicate files before uploading to server using the selected event.

Compare the selected files with the existing files data and filter the file list by removing the duplicate files.

**CSHTML**

```
@Html.EJS().Uploader("UploadFiles").Selected("onFileSelected").AutoUpload(false).AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
<script>
 function onFileSelected(args) {
 var isNullOrUndefined = ej.base.isNullOrUndefined;
 let existingFiles = this.GetFilesData();
 for (i = 0; i < args.filesData.length; i++) {
 for (j = 0; j < existingFiles.length; j++) {
 if (!isNullOrUndefined(args.filesData[i])) {
 if (existingFiles[j].name == args.filesData[i].name) {
 args.filesData.splice(i, 1);
 }
 }
 }
 }
 existingFiles = existingFiles.concat(args.filesData);
 args.modifiedFilesData = existingFiles;
 args.isModified = true;
 }
</script>
```

**GETTING-STARTED.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

See Also

- [Validate image/\\* on drop](#)
- [Determine whether uploader has file input \(required validation\)](#)
- [Check file size before uploading it](#)
- [Check the MIME type of file before uploading it](#)

## Forms Support (Synchronous Upload)

The Uploader control works with HTML form like default file input. The following configuration is must to make the Uploader work inside the form.

- `saveUrl` and `removeUrl` must be null.
- `autoUpload` must be disabled.
- `name` attribute must be added in input element.

The selected or dropped files are received as a collection in form action when the form is submitted. The form action handles the server-side operations that manage the file upload process. When you reset the form, the file list and data will be cleared.

### CSHTML

```
<div class="col-lg-12 control-section">
 <h4 class="form-title">Photo Contest</h4>
 <div class="control_wrapper" id="control_wrapper">
 <!-- Initialize Uploader -->
 <form id="form1" method="post">
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <input type="text" id="name" name="name" data-required-
message="* Enter your name" required="" data-msg-containerid="nameError">

 <label class="e-float-text e-label-top"
for="name">Name</label>
 </div>
 <div id="nameError"></div>
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <input type="email" id="email" name="email" data-
validation="email" data-required-message="* Enter your email" required=""
data-msg-containerid="mailError">

 <label class="e-float-text e-label-top"
for="email">Email</label>
 </div>
 <div id="mailError"></div>
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <input type="text" id="mobilen0" name="mobile" data-
required-message="* Enter your mobile number" required="" data-msg-
containerid="noError">

 <label class="e-float-text e-label-top"
for="mobile">Mobile No</label>
 </div>
 <div id="noError"></div>
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input upload-area">
 <input type="text" id="upload" name="upload" readonly
data-required-message="* Select any file" required="" data-msg-
containerid="uploadError">
 <button id="browse" type="button" class="e-control e-btn
e-info" onclick="browseClick()">Browse..</button>

 </div>
 </div>
 </form>
 </div>
</div>
```

```

 <label class="e-float-text e-label-top"
for="upload">Choose a file</label>
 </div>
 <div id="uploadError"></div>

@Html.EJS().Uploader("fileupload").AutoUpload(false).Selected("onFileSelect"
).Multiple(false).AllowedExtensions("image/*").Render()
 </div>
 <div class="form-group" style="padding-top: 11px;">
 <div class="e-float-input">
 <textarea class="address-field" rows="4"
id="Address"></textarea>

 <label class="e-float-text e-label-top">Address</label>
 </div>
 </div>
</form>
<div class="submitBtn">
 <button class="submit-btn e-btn" id="submit-btn"
onclick="onFormSubmit()">Submit</button>
</div>

@Html.EJS().Dialog("confirmationDialog").Width("335px").Visible(false).Conte
nt("Your details has been updated successfully, Thank
you").Target("#control_wrapper").IsModal(true).AnimationSettings(new
Syncfusion.EJ2.Popups.DialogAnimationSettings() { Effect =
Syncfusion.EJ2.Popups.DialogEffect.Zoom }).Render()
 </div>
</div>
<script>
 window.onload = function () {
 var confirm =
document.getElementById("confirmationDialog").ej2_instances[0];
 confirm.visible = false;
 inputElement = document.getElementById('upload');
 formID = document.getElementById('form1');
 formObj = new ej.inputs.FormValidator(formID, options);
 }
 function onFileSelect(args) {
 var inputElement = document.getElementById('upload');
 inputElement.value = args.filesData[0].name;
 }
 var options = {
 customPlacement: function (inputElement, errorElement) {
 inputElement = inputElement.closest('.form-
group').querySelector('.error');
 inputElement.parentElement.appendChild(errorElement);
 },
 rules: {
 'name': {
 required: true
 },
 'email': {
 required: true
 },
 'upload': {
 required: true
 }
 }
 }

```

```

 },
 'mobile': {
 required: true
 }
 }
};
inputElement = document.getElementById('upload');
formID = document.getElementById('form1');
var formObj = new ej.inputs.FormValidator(formID, options);
function onFormSubmit() {
 var confirm =
document.getElementById("confirmationDialog").ej2_instances[0];
 var formStatus = formObj.validate();
 if (formStatus) {
 formObj.element.reset();
 confirm.show();
 }
 confirm.overlayClick = function () {
 confirm.hide();
 };
}
function browseClick() {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
 return false;
};
</script>

```

## INDEX.CSS

```

.control_wrapper {
 max-width: 400px;
 margin: auto;
}
#control_wrapper {
 max-width: 500px;
 margin: auto;
 border: 0.5px solid #BEBEBE;
 box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
 padding: 1% 4% 2%;
 background: #f9f9f9;
}
.upload-area {
 width: 73%;
}
.e-bigger .upload-area {
 width: 68%;
}
.e-error {
 padding-top: 3px;
}
.e-upload {
 width: 100%;
 position: relative;
 margin-top: 15px;
}

```

```
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}
.submit-btn {
 margin-top: 15px;
 position: relative;
}
.submitBtn .desc {
 margin: 2% 23% 0% 18%;
}
@media only screen and (max-width: 500px) {
 .submitBtn .desc {
 margin: 12px;
 }
 .upload-area, .e-bigger .upload-area {
 width: 60%;
 }
}
.submitBtn {
 position: relative;
 text-align: center;
}
.form-support {
 width: 100%;
}
.success .form-support {
 display: none;
}
.success .successmsg {
 border: 0.5px solid green;
 padding: 10%;
 color: green;
}
form#form1 {
 position: relative;
 top: 14%;
}
.form-support td {
 width: 100%;
 padding-top: 4%;
}
.e-upload {
 display: none;
}
input.choose-file {
 width: 60%;
}
#mobile-no input[type=number]::-webkit-inner-spin-button,
#mobile-no input[type=number]::-webkit-outer-spin-button {
 -webkit-appearance: none;
 margin: 0;
}
button#browse {
 float: right;
 position: relative;
 margin-right: -113px;
 margin-top: -27px;
}
```

```

}
@media only screen and (max-width: 600px) {
 .submitBtn {
 margin-top: -22px;
 }
}
.material button#browse {
 margin-right: -117px;
}
.form-title {
 text-align: center;
}

```

## Template

You can customize the default appearance of the uploader using a template along with buttons.

### File list template

The [template](#) property is used to customize the default appearance of each file in the list. It can be represented as the HTML element or string. The selected or dropped files are displayed as per the template layout provided. The remove and progress bar action is handled using the corresponding events when the template is defined.

For example, you can display file type icon along with the default UI elements.

### CSHTML

```

<div id="container">
 <div id='dropArea'>
 Drop files here or <a href=""
id='browse'><u>Browse</u>

@Html.EJS().Uploader("UploadFiles").Success("onuploadSuccess").DropArea("#dr
opArea").Failure("onuploadFailed").Selected("onSelect").Progress("onFileUplo
ad").Template("<span class='icon sf-icon-
${type}'>${name}" +
 "${size} bytes
 " +
 "<progress id='progressBar' class='progressbar' value='0'
max='100'></progress> <span class='percent-td
percent'>").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove"
}).AutoUpload(false).Render()
 </div>
</div>
<script>
 document.getElementById('browse').onclick = function () {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
 return false;
 };
 function onFileUpload(args) {
 let li = this.uploadWrapper.querySelector('[data-file-name="' +
args.file.name + '"]');

```

```

 let progressValue = Math.round((args.e.loaded / args.e.total) *
100);
 li.getElementsByTagName('progress')[0].value = progressValue;
 li.getElementsByClassName('percent')[0].textContent =
progressValue.toString() + " %";
 }
 function onuploadSuccess(args) {
 if (args.operation === 'remove') {
 let height = document.getElementById('dropArea').style.height;
 height = (parseInt(height) - 40) + 'px';
 document.getElementById('dropArea').style.height = height;
 } else {
 let li = this.uploadWrapper.querySelector('[data-file-name="' +
args.file.name + '"');
 let progressBar = li.getElementsByTagName('progress')[0];
 progressBar.classList.add('e-upload-success');
 li.getElementsByClassName('percent')[0].classList.add('e-upload-
success');
 let height = document.getElementById('dropArea').style.height;
 document.getElementById('dropArea').style.height =
parseInt(height) - 15 + 'px';
 }
 }
 function onuploadFailed(args) {
 let li = this.uploadWrapper.querySelector('[data-file-name="' +
args.file.name + '"');
 let progressBar = li.getElementsByTagName('progress')[0];
 progressBar.classList.add('e-upload-failed');
 li.getElementsByClassName('percent')[0].classList.add('e-upload-
failed');
 }
 function onSelect(args) {
 let length = args.filesData.length;
 let height = document.getElementById('dropArea').style.height;
 height = parseInt(height) + (length * 55) + 'px';
 document.getElementById('dropArea').style.height = height;
 }
</script>

```

## INDEX.CSS

```

.e-upload {
 width: 100%;
}
.e-file-select-wrap {
 display: none;
}
.e-upload {
 border: none;
 margin-top: 15px;
}
#drop {
 padding-left: 30%;
 font-size: 14px;
 font-family: "Roboto", "Segoe UI", "GeezaPro", "DejaVu Serif", "sans-
serif"

```



```

}
#dropArea {
 min-height: 18px;
 border: 1px dashed #c3c3cc;
 padding: 15px 0;
 margin: 0 auto;
 width: 440px;
}
.wrapper{
 padding: 0 10px;
}
.e-upload-success, .e-upload-failed {
 display: none;
}
.sf-icon-bmp:before { content: "\e700"; }
.sf-icon-xlsx:before, .sf-icon-XLSX:before { content: "\e701"; }
.sf-icon-avi:before, .sf-icon-AVI:before { content: "\e702"; }
.sf-icon-doc:before, .sf-icon-DOC:before { content: "\e703"; }
.sf-icon-exe:before { content: "\e704"; }
.sf-icon-mp4:before { content: "\e705"; }
.sf-icon-zip:before, .sf-icon-ZIP:before { content: "\e706"; }
.sf-icon-tar:before { content: "\e707"; }
.sf-icon-docx:before { content: "\e708"; }
.sf-icon-jpg:before { content: "\e709"; }
.sf-icon-png:before { content: "\e70a"; }
.sf-icon-gif:before { content: "\e70b"; }
.sf-icon-pdf:before { content: "\e70c"; }
.sf-icon-txt:before { content: "\e70d"; }
.sf-icon-jpeg:before { content: "\e70e"; }
.sf-icon-xls:before { content: "\e70f"; }
.sf-icon-mp3:before { content: "\e710"; }
#dropArea .e-upload .e-upload-files .e-file-delete-btn,
#dropArea .e-upload .e-upload-files .e-file-remove-btn {
 top: 35%;
}
span.file-icon{
 width: 25px;
 height: 25px;
 display: inline-flex;
 background-size: contain;
 margin: 7px;
}
img.file-icon {
 width: 20px;
 height: 20px;
}
.file-name {
 color: #e1e1e1;
 font-size: 14px;
 padding: 3px 10px;
 overflow: hidden;
 text-overflow: ellipsis;
 display: inline-block;
 width: 50%;
 white-space: nowrap;
 position: relative;
 top: 5px;
}

```

```

}
.file-size {
 font-size: 12px;
 padding: 3px 10px;
 overflow: hidden;
 display: inline-block;
 position: relative;
 top: 6px;
}
}
.progressbar{
 height: 5px;
 width: 70%;
 margin-left: 14px;
}
}
.upload-success{
 background-color: green
}
.upload-failed{
 background-color: red;
}
}
#dropArea .e-upload .e-upload-files .e-upload-file-list {
 min-height: 63px;
}
}
#dropArea .e-upload {
 border-width: 0 1px;
}
}
/* csslint ignore:start */
@font-face {
font-family: 'FileType_Font';
src:
url(data:application/x-font-ttf;charset=utf-
8;base64,AAEAAAAKAIAAAwAgTlMvMjltSgIAAAEoAAAAVmNtYXDNyOfNAAABYAAAAFZnbHlmBIu
kdGAAAKgAAB78aGVhZA/Zy4MAAADQAAAAANmhoZWEHlAQTAaaaRaaaACRobXR4SAAAAAAYAAAAAB
IbG9jYUgKP2YAAAIgAAAAJmlheHABKQFJAAABCAAAACBuYW1lPdPGuQAAIUQAAAJtcG9zdGQHrBs
AAC00AAAAkaABAAEAAAAAFwEAAAAAADdwABAAAAAaAAAAAAAAAAAAAAAEgABAAAAQAavWuGjl8
PPPUACwQAAAAAANaiQ0kAAAAA1qJDSQAAAAADdwP0AAAACAACAAAAAaAAAAAASAT0ADQAAAAA
AAgAAAAoACgAAAP8AAAAAaAAAAQAAZAABQAAAokCzAAAAI8CiQLMAAAB6wAyAQgAAAIABQMAAAA
AAAAAAAAAAAAAAAAAAAAAUGZFZABA5wDnEAQAAAAAXAQAAAAAaBAAAAAaBAAAAAQAAAA
EAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQAAAAEAAAABAAAAAQ
AAAAEAAAAAAAAAagAAAAMAAAAUAAMAAQAAABQABABCAAAABAAEAAEAOCQ//8AA0cA//8AAAABAAQ
AAAABAAIAAAwAEAAUABgAHAAGACQAKAAsADAANAA4ADwAQABEAAAAAAAAABCGHaAswDrAQ4BNQFsAZ
iB8wIsAlKcsgMAAYEDbgOrg9+AAAACAAAAAADdwP0ABIAJQA4AFgAZwCRANIA3QAAATM/Bj0BLwY
jJTM/Bj0BLwUrAQU7AT8GLwcjJRczHwsVDwsjFSM1Ixc3MxUjNTcHIycXFSM1IzMfChUPBh8HDwo
jNQMRFR8NMyEzPw01ESSBLwg9ASEjDw0FHwcZJwEjJQcGBgUDAwICAgMEBgYHJwGOJwGHBgUEAwI
CAwQFBgcHKP5yIAGGBQUEAwEBAQEDAwUFBwggAbUHCA0GBgUFBAQDAwIBAQEBAgMDBAQFBQYMDy4
hvTc2KiEDNxc4BCFmCAcNCwoDBAMCAGEBAGIDBAQGBgcGBgQEAwEBAQIBAwMDBAUkDA5MegICAwQ
FBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAGH
0AQEDBAUFBgbb+gE/AQEDAwQFBgYHBgUEAwMBAQsBAQMDBQUHBwcGBgUEAwI0AgIDBAQFBgcFBQM
DAgEBGgEDAwIEBAQEYFBgYHDQYFBgQFBAMDawQCRryQkLw+U5GRUz68AQIEBgMEBAUFBQwGBgY
FBQQAawMDBQUGBgcIBwsFBQUEBAMFBAK8AbX81AoJCQkICAcHBgUFBAMCAGICAwQFBQYHBwgICQk
JCgJSAQEFBwgKCwYHBvoCAGMEBQUGBwcICAKJCYcGBgUFBaICAfoAAAYAAAAA3cD9AAZADMATQB
ZAGQApQAAATMfAx0BDwMjISMvAz0BPwMzJTMfAx0BDwMrAi8DPQE/AzM3Mx8DHQEPAysCLwM9AT8
DMYcXNZMHfYmNByM3JwEfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AUoDAwUEAQEEBQMD/iwEAwQ
EAQEEBAMEAdQDAwUEAQEEBQMD2gQDBAQBAQQEAwTaAwMFBABBAUDA9oEAWQEAQEEBAME1SssJ0B
CJy4tKENBAW0BAQMEBQUGBtv6/gwCAGMEBQUGBwcICAKJCQoCMgoJCQkICAcHBgUFBAMCAvoGBwY
LCggHBQEB/qgKCQkJCAgHBwYFBQQDAgIBBgEEBQMDAwMFAwEBawUDAwMDBQQBfQEEBQMDAwMFAwE
BAwUDAwMDBQQBfQEEBQMDAwMFAwEBawUDAwMDBQQBCUREXF9GR19cARAGBgUFBAMBAfpe/NQKCQk

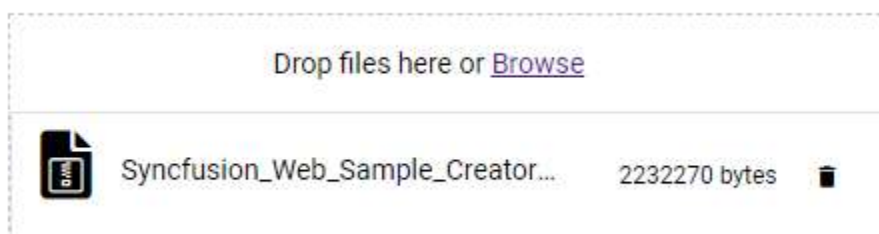
```

JCAgHBwYFBQQDAgICAgMEBQUGBwcICAKJCQoCUgEBBQcICgsGBwb6AQECaWQFBQcGBwgICQkJAAA  
 ABQAAAAADdwP0AAIAQwCEAI8A0AAAAQc1BxUfDz8OPQEvDg8OBQcVDw4vDz8PHw4DHwcZJwURFR8  
 NMyEzPw01ESsBLwg9ASEPDgJefX0BAGQGBwkKCwwNDw8PERESERERDw8PDQwLCgkHBgQDAwQGBwk  
 KCwwNDw8PEREREhERDw8PDQwLCgkHBgQCAXYBAwUHCQoMDQ4QERETFBQUFRQUEhIREA4NDAoJBgU  
 DAgIDBQYJCgWNDhAREhIUFBUFFBQTEREQDg0MCgkHBQNDaQEDBAUFBgbb+v4MAgIDBAUFBgCHCAg  
 JCQkKAjIKCQkJCAgHBwYFBQQDAgL6BgCGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICAXNOnU8JCBE  
 REA8ODQwLCgkHBgQDAQEDBAYHCQoLDA0ODxARERESERAQDw4ODAsKCAgFBQIBAQIFBQgICgsMDg4  
 PEBAREgoKFRMTEREQDg0MCgkHBQMBAQMFBwkKDA0OEBERExMVFBUFFBISEQ8PDQwKCAcFAwEBAwU  
 HCAoMDQ8PERISFBQBkQYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJ  
 SAQEFBwgKCwYHBvoBAQIDBAUFwYHCAGJCQkAAAYAAAAA3cD9AAZADMATQBmAHEAsgAAATMfAx0  
 BDwMjISMvAz0BPwMzJTMfAx0BDwMrAi8EPwQzNzMfAx0BDwMrAi8EPwQzJR8CMz8CMx8BMz8BMwC  
 jLwIXDwEjJyUfBzMNbREVHw0zITM/DTURKwEvCD0BIQ8OAwODAwUDAQEDBQMD/gwEAWQEAEQBAM  
 EAfQDAwUDAQEDBQMDvAMDBQMBAQEBAwUDA7wDAwUDAQEDBQMDvAMDBQMBAQEBAwUDA/7AJwEACAE  
 DKXsnAwEBKR04HycCAQIShjkBjgEBAwQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwc  
 GBQUEAwIC+gYHBgsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgElAQMFawMEAgUEAQEEBQIEAwMFAwF  
 9AQMFawMEAgUEAQEEBQIEAwMFAwF9AQMFawMEAgUEAQEEBQIEAwMFAwEBhgMNCAMfhhAHj7J/CAk  
 HibL5BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAgJCQkKALIBAQUHCAoLBgc  
 G+gEBAgMEBQUHBgcICAKJCQAAAAFAAAAAAN3A/QACwAXACMALgBvAAABFSVMvUjFTMVIzUjFzc  
 zBxcjJwcjNycjFSMVMvUjFTMVIzUBHwcZJwURFR8NMyEzPw01ESsBLwg9ASEPDgLYWU1NWnuOKCc  
 mOTomKCkmOzkPWU1NWnsBWwEBAwQFBQYG2/r+DAICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQU  
 EAwIC+gYHBgsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgHhGzMaOhq8RERdX0VFX10bMx06GrwBOAY  
 GBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQFBQYHBwgICQkJCgJSAQEFBwgKCwYHBvoBAQI  
 DBAUFwYHCAGJCQkAAAAACAAAAAADdwP0AAQACAAMABEAFQAxADwAfQAAARUzNSMFMzUjFzM1IzM  
 VMzUjBTM1IzUVMzUzFTM1MxUzNTMRIzUjFSM1IxUjNSMVIxElHwcZJwURFR8NMyEzPw01ESsBLwg  
 9ASEPDgJ9Pj7+xz8/Xry82z4+/sc/Pz8fvB8/Hx8/H7wfPx8BWAEBawQFBQYG2/r+DAICAwQFBQY  
 HBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgsKCAcFAQH+qAoJCQkICAcHBgUFBAMCAgElH15  
 eXj+cPl5eXj4fHz4+Hx/+qCagPz8gIAFY+gYGBQUEAwEB+1781AoJCQkICAcHBgUFBAMCAgICAwQ  
 FBQYHBwgICQkJCgJSAQEFBwgKCwYHBvoBAQIDBAUFwYHCAGJCQkAAAAABwAAAAADdwP0AAMABwA  
 jAEcAaACpALQAACUzNSM3FSM1NxUjFTMVIxUzFSMVMvUjNSM1MzUjNTM1IzUzNScRHwYzITM/BhE  
 vBjEPBiUfBxEPBiEvBhE/BgMRFR8NMyEzPw01ESsBLwg9ASEjDw0FHwcZJwHhPj5dfX0+Pj4+Pj4  
 +Pz8/Pz8/2wEBAgIDAwQEAY4EBAMDAGIBAQBAGIDAwQE/nIEBAMDAGIBAAEFcGkIBwYEAgiEBGg  
 ICQr+aaAoJCACGBAICBAYHCAkKqWICAwQFBQYHBwgICQkJCgIyCgkJCQgIBwcGBQUEAwIC+gYHBgs  
 KCAcFAQH+qAoJCQkICAcHBgUFBAMCAgH0AQEDBAUFBgbb+ucfH1ld+h8fIB8fHyAgHx8gHx8fDP5  
 yBAQDAwMBAgIBAwMDBAQBJgQDBAMCAgEBAQEACgMEAY8BAGMGBwgJCv5oCgkJBwUEAgIEBQcJCQo  
 BmAoJCACGBAIBOPzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAgJCQkKALIBAQUHCAoLBgcG+gI  
 CAwQFBQYHBwgICQkJhwYGBQUEAgIB+gAAAAHAHAHAAN3A/QAAGAVADMAOWBDAE4AjwAAATMnFzs  
 BPwU9AS8FKwE3HwwPBxcVIycjFSM1IxcjJyMHIZcjFSMVIzUjNQEfBzMNbREVHw0zITM/DTURKwE  
 vCD0BIQ8OAdw2G4whCAGBQQDAgICBAUGBwgiIQgODQUBFQQEAWICAgEBAQMEBQYHCCsjJiQhXEc  
 iD0kPiKc/OyA6AWgBAQMEBQUGBtv6/gwCAGMEBQUGBwcICAKJCQoCMgoJCQkICAcHBgUFBAMCAVo  
 GBwYLCgHQBEB/qgKCQkJCAgHBwYFBQQDAgIBa04xAgIEBAYGBwcGBQUdAwIbAQEEAwMDBAQFBQU  
 GDQkICAcGBQQUETgJISLy8LCy8G6GhGwE4BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAU  
 FBgCHCAgJCQkKALIBAQUHCAoLBgcG+gEBAgMEBQUHBgcICAKJCQAAAAgAAAAA3cD9AAVADgARAB  
 fALYA5gDxATIAABMzPwg1LwgjFw8EHwc/CC8IDwIlFzcZBxcjJwcjNychHwsPCyM1BR8GIy8HIw8  
 GFR8HMz8GMw8NIy8KNT8KMx8EJx8GHQEPcSsBLwK9AT8JOWEfARMfBzMNbREVHw0zITM/DTURKwE  
 vCD0BIQ8O5RUMCwUEBACFAwIBBAUGBAQFBQsdsgIFAwEBAgMEBgECQoKCQQDBwUEAgEBAgQFBwM  
 ECQsKCAgBTicoJTg6JikoJjo5/jYMCwsKCQgHBgQDAgEBAQOEbgICgoLCw02AdwEBAQDAgIDIQE  
 DAwUFBgIBgkIBwYEAwIBAgMFBggICgkIBgYEBaICIQMCAgMEBAUFBQYGBwCPCwoKCQgHBgUEAwI  
 BAQIDBAUGBwkJCgoLDw4GBQa6CAGGBQUdAgIDBAYGBwCJCgoLCwsKCQgIBgUFaWICAwGBgGgICQo  
 KCwwKCqUBAQMEBQUGBtv6/gwCAGMEBQUGBwcICAKJCQoCMgoJCQkICAcHBgUFBAMCAVoGBwYLCg  
 HBQEB/qgKCQkJCAgHBwYFBQQDAgIBPwEDAgMECAkMDRkOCwoIAwMCAgIOBQoLDhONDAoJBgIEAQE  
 DAwMHCQsNDxgNCwoHAWIDAQEDBSJERf1fRUVfXQECBAQHbwgKCgWMDRYMCwsJCQcGBQMCAbwPBAU  
 FBgYHDgkIBwUEAwEBAgQGCAkMDRgODAsJBwUDAQIDBAUGCAkOBgYGBQUFBAMDAGIBAQIEBAYHCAk  
 LCwsNEQ0MDAOKCAcGBQQCAQMCAwQCBgcJCQsMDA4XDQwKCgkHBgUEAgIEBQYHCQoKDAwNGAwMCwo  
 ICAYFAwMDAwE8BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAgJCQkKALIBAQU  
 HCAoLBgcG+gEBAgMEBQUHBgcICAKJCQAGAAAAAN3A/QABAaKAEcAbAB3ALgAAAEjNxc/ARUPBi8  
 GPQE/Bh8GJREfByE/BxEvBiMhIw8EJRczHwYRDwchLwCRPwclHwcZJwURFR8NMyEzPw01ESsBLwg  
 9ASEPDgJ9+ipTU0kCAwQFBQYGBgYGBAQDAgIDBAQGBgYGBgUFBAMC/qgBAQECAwIDBAF1BAMDAGI

BAQEBAQECAgMDBP6LBAMCBQEBAYUFBQkIBwYCBAlBAwQGBwQICv6GCgkIBwYCBAlBAwQGBwQJCQE  
tAQEDBAUFBgbb+v4MAgIDBAUFBgCHCagJCQkKajIKCQkJCAGHBwYFBQQDAgL6BgGCwoIBwUBAf6  
oCgkJCQgIBwcGBQUEAwICAQZ1TnU/BwUGBAQDAgEBAGMEBAYFBwYGBQUEAwEBAQEEDBAUFbkf+qgQ  
DAgMCAQEBAQEBAgMCAwQBVgMEAgMCAQEBAQUdAY0BAwQGBwQJCf6lCgkIBwYCBAlBAwQGBwQJCQF  
bCgkIBwYCBAG8BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCagJCQkKAlIBaQU  
HCAoLBgcG+gEBAGMEBQUHBgcICakJCQAAAAAAN3A/QAAwAHAAsADwAzAD4AfwAAATM1IzsBNSM  
HMzUjOwE1IycVMzUzFTM1MxUjFTMVIxUzFSM1IxUjNSMVIzUzNSM1MzUjNSUfBzMnBREVHw0zITM  
/DTURKwEvCD0BIQ80AeE+Pj4/P30/Pz8+Pj8/Pj8+Pj4+Pj4/Pj8+Pj4+PgEZAQEEDBAUFBgbb+v4  
MAgIDBAUFBgCHCagJCQkKajIKCQkJCAGHBwYFBQQDAgL6BgGCwoIBwUBAf6oCgkJCQgIBwcGBQU  
EAWICASU/Pj4+Pz4+Pj4+Pj8+Pz4+Pj4+Pj8+Pz76BgYFBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwI  
CAGIDBAUFBgCHCagJCQkKAlIBaQUHCAoLBgcG+gEBAGMEBQUHBgcICakJCQAAAA0AAAAA3cD9AA  
RACMANgBJAFMAVwCnAlkAywDdAPAA+wE8AAALDwc1PwcHFS8HNx8GJQ8IJz8HIR8IBY8HJRujFTM  
VIxUjNSMVIzUHHwQjLwYrAQ8GFR8HMz8CNSM1MxUPBy8LNT8OHwIFIY8HNx8GJQ8HIz8HJQcvBzU  
fBicjDwcnPwgfBzMnBREVHw0zITM/DTURKwEvCD0BIQ80AqYQEBITFBQVFRISEhARDw8NoBUVFBQ  
TEhAQFA4PDxAREhIBGwEBBAYHCQoMDRULCgkIBGUEAv5GAQEEBQYICQoLFQ0MCgkHBGQCAaJOREQ  
dJB01AwQGBAlcAgIDBAUGBgFCQCHBQQCagECBAUGBwkKCgkHBSI/BQUHCAgJCQoLCQkJBwcGBQQ  
DAQEBAgICAwQEBQUGBgCHBwGNDAsBPB4CAwUGCAkKCxUNDAoJBwYE/mILCgkIBGUEAh0CBAYHCQo  
MDQFhFQ0PDxAREhISFRUUFMBSEKUJCRISERAPDw0VEBASExQUFRWMAQEEDBAUFBgbb+v4MAgIDBAU  
FBgcCHCagJCQkKajIKCQkJCAGHBwYFBQQDAgL6BgGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICyA0  
MCgkHBGQCHgIDBQYICQoLNx0CBAYHCQoMDRUMCggIBGUEzAsKFRQUEXIQEBUNDw8QERISEGkJEH  
REA8PDRUQEBITFBQVFU8XMhdHp6e0DAQECQsMBwYFBQMCAGIDBQgICwwTDQsKCAcFagECawUhFkE  
FBQQEAgIBAQECAwQFBwCICgoKDBYJCAGHBwYFBQUdAwMBAQEBAwQsEhISEBEPDw0VEBERExQUFW0  
NDw8REBISEhUVFBQTEREQFRULCgkIBGUDAx0CBAYHCQoMFAIDBQYICQoLFQ0MCgkHBGQCNAYGBQU  
EAWEB+1781AoJCQkICACHBgUFBAMCAGICAwQFBQYHBwGICQkJCgJSAQEFBwGKCwYHBvoBAQIDBAU  
FBwYHCAgJCQkAAAAABwAAAAADdwP0ABUALAA3AE0AuQDEAQUAAAEpBhUfBj7GHwQ7AT8FNS8GByc  
PAT8DLwQxDwQdAR8CPwI1LwUfAhUPAh8DPwIfCh0BDwsjLwUPDC8JPw4vBT8JMx8GNx8HMyCfERU  
fDTMhMz8NNRErAS8IPQEHdW4BgQsVDgYEAwECAQMDAwQDBAQDBQca5AkUCAKEBAUFBQQCAGEBawM  
EBhEKEZsNDxozGGRgQEBAPHAUFBAIBAw8OCAMBAQICAgCHMGIBAQISFRYXGBYVfGSLDQoIBwUDAwE  
BAQIDAwMEBQUFBgUGBgkICAcNMDIfICERHACFBQYHCAGIBwCHBgUFAwMBAQIFBwGJCQlGCwoKCAg  
HFhUEAgMBAQMFAwQFBQYFBwoICAGHBwUEEQAEBwQFBQYG2/r+DAICAwQFBQYHBwGICQkJCgIyCgk  
JCQgIBwcGBQUEAwIC+gYHBGsKCAcFAQH+qAoJCQkICACHBgUFBAMCAGElBAgIBAQFBQMFAwMDAGe  
BAQEBAOuPactBwUCAGMEBQQFBQMEAGICAgEBPiQiCAYGBRESEhOZAgIEBAUFBQkZFR8MDAYFBAM  
DBAIGCQgJCQhDGxozGAQDAQEBAgQFBQYFBgYGBGUGBgUFBQUEBAMCAGEBAGMEciwJCAkLHzELBQQ  
EAWIBAQEDBAQGBgcKCgkJCQCHBgUEGxcXGBGxGSEhCgUKCwoKCQUEAwQDAgEBAGMFBQYHvQYGBQU  
EAWEB+1781AoJCQkICACHBgUFBAMCAGICAwQFBQYHBwGICQkJCgJSAQEFBwGKCwYHBvoBAQIDBAU  
FBwYHCAgJCQkABQAAAAADdwP0AAcAEwAbACYAZwAAARUjFSM1IzUjFzczBxcjJwcjNycjFSMVIzU  
jNQEfbzMnBREVHw0zITM/DhErAS8JNSEjDw0C8DogOoEoJyY5OiYoKSY6OAw7IDoBaAEBAwQFBQY  
G2/r+DAICAwQFBQYHBwGICQkJCgIyCgkJCQgIBwYHBQUEAwIBafoGBwYLCgGHBQEBAf6pCgkJCQg  
IBwcGBQUEAwICAEBoaEbRERdX0VFX10boaEbATgBgGUGBAMBAfpe/NQKCQkJCAGHBwYFBQQDAgI  
CAGMEBQUGBwCICakJCQoCUgEBBQcICGsGBwb6AgIDBAUFBgCHCagJCQkAAAAABwAAAAADdwP0ABI  
AHgA+AGcAuWDGAQCAAAEzPwY9AS8FKwE1FSMVMxUjFTMVIzUjFzMfCxUPCyMVIzUjFQ8KLwkzFR8  
FOWE/BT0BBR8FIY8GKwEPBh0BHwczPwI1IzUzFQ8GKwEvCjU/DTsBHWMDHwczJwURFR8NMyeZPw0  
1ESsBLwG9ASEjDw0BdycJBwYFBAMCAGMEBQYHCCgBa1lNTVp7YQgHDgYFBGQFAwQCAgIBAQICAgM  
EBAUFBg0OLyAlAQICAgMDBAkLCw4NDQoFCAMDAGMBIQEDAwQFBGyHBQUEBAICaf0EBAQDBAMgAgI  
EBAYGBwGQCQkHBQUCAgMEBQCEBQkLDAkIBYdHBGyICakKCwsLCwoKCAgGBGQDAgEBAGIDAwUEBgY  
GBwCICakIDgWGBYwBAQMEBQUGBtv6/gwCAGMEBQUGBwCICakJCQoCMgoJCQkICACHBgUFBAMCAvo  
GBwYLCgGHBQEBAf6pCgkJCQgIBwYHBQUEAwIBafoGBwYLCgGHBQEBAf6pCgkJCQgIBwYHBQ  
EBAUGLQYGDgYGBGUEBQDBAIEAka8hAYMBgUFBAGUBQMBQIFAwcFBAULDQcGBGQDAgICAwMFBGy  
HhA0EBQUFBA4IBwYFAwMCAgQGAoLDhUPDAsJCAMCAwEDAwULGECHBQUEAwIBAGMEBQCEBQJCQoLDA0  
ZCgkJCACHBgYFBAMDAgECBAIEAUIGBgUFBAMBAfpe/NQKCQkJCAGHBwYFBQQDAgICAgMEBQUGBwC  
ICakJCQoCUgEBBQcICGsGBwb6AgIDBAUFBgCHCagJCQkAAAAUAAAAA3cD9AAAFABEAggCNAM4AAAE  
VMxUjNSMXNzMHFyMnByM3JwUfBhUjLwYrAQ8FHQEFDRUPCiMvCjMVHwU7AT8GLw89AT8JMx8BAx8  
HMyCfERUfDTMhMz8NNRErAS8IPQEHdW4B6lV2jignJjk6JigpJjs5AaMHBwYEBAMCIAECAGUFBgc  
IBwcGBQQCAGIDBA0mCAGGBQUdAwIBAQECAwMEBAoLDQ8JCgkJCACGBQUdAQEHAgQEBGcICQgHBG  
EAgIBAQICBAMJKAgIBGUEBAMCAQEBAgMDCAoLDQ4KCQg0AQEDBAUFBgbb+v4MAgIDBAUFBgCHCag  
JCQkKajIKCQkJCAGHBwYFBQQDAgL6BgGCwoIBwUBAf6oCgkJCQgIBwcGBQUEAwICAGiGrxERF1  
fRUVfXQUEBQUGBwGHCACGBQUEAgIBAwIEBQUFBQUEBACNBAQEBAQFBGyGBwsFBQUEBAMHBAIBAgI

```
DBAUGBgCHCAkIBgYFBAICAQIDBAQFBgUGBAQBBQ4EBAUEBQUFBgYGBgUFBQUECAYFAGECAGe6BgY
FBQQDAQH6XvzUCgkJCQgIBwcGBQUEAwICAgIDBAUFBgCHCAgJCQkKAlIBAQUHCAoLBgcG+gEBAGM
EBQUHBgcICAKJCQAAAAQAAAAA3cD9AADAGIAbQCuAAABFTc1Nx8CFREVDwkjLwk9AT8KOWEXNQc
VDwkrAS8IPQE/CjsBFz0BPwU7ARcnHwcZJwURFR8NMyEzPw01ESsBLwg9ASEPDgGvxx8DAgICAwQ
ICQsMCAkICQkJCAcHBgQEAgECAQMDCaKLDagJCAkICcMCAwQICQsMCAkICQkJCAcHBgQEAgECAgI
EBwKLDagJCAkJCAEEBQMD6wQDBxUBAQMEBQUGBtv6/gwCAgMEBQUGBwCICAKJCQoCMGoJCQkICAc
HBgUFBAMCAVoGBwYLCggHBQE/qqKCQkJCAgHBwYFBQQDAgIB7CQoJCYDBAUE/tIICQgICwKHBwM
CAgEBAwQFBgYIBQcGBgYHBgYKCQgGAWMBAp0n1QkJCQoJCAYDAgICAwQFBQcHBgYHBgYGBwYKCQg
GAWICAvGEAwYFAGEvAt0GBgUFBAMBAfpe/NQKCQkJCAgHBwYFBQQDAgICAgMEBQUGBwCICAKJCQo
CUgEBBQcICGsGBwB6AQECawQFBQcGBwgICQkJAAAAAASAN4AAQAAAAAAAAABAAAAQAAAAAAQA
NAAEAQAAAAAAGAHAA4AAQAAAAAAAwANABUAAQAAAAAABAANACTIAAQAAAAAABQALAC8AAQAAAAA
ABgANADoAAQAAAAAAGcAsAEcAAQAAAAAACwASAHMAAwABBakAAAACAIUAawABBakAAQAaAICAAwA
BBakAAgAOAKEAAwABBakAAwAaAK8AAwABBakABAAaAMkAAwABBakABQAWAOMAawABBakABgAaAPk
AAwABBakACgBYARMAAwABBakACwAkAWsgRmlsZVR5cGVfRm9udFJlZ3VsYXJGaWxlVHlwZV9Gb25
0RmlsZVR5cGVfRm9udFZlcnNpb24gMS4wRmlsZVR5cGVfRm9udEZvbnQgZ2VuZXJhdGVkIHVzaW5
nIFN5bmNmdXNpb24gTWV0cm8gU3R1ZGlvd3d3LnN5bmNmdXNpb24uY29tACAARgBpAGwAZQBUAHk
AcABlAF8ARgBvAG4AdABSAGUAZwBlAGwAYQByAEYAaQBsAGUAVAB5AHAAZQBfAEYAAbwBuAHQARgB
pAGwAZQBUAHkAcABlAF8ARgBvAG4AdABWAGUAcgBzAGkAbwBuACAAMQAuADAARgBpAGwAZQBUAHk
AcABlAF8ARgBvAG4AdABGAG8AbgB0ACAAZwBlAG4AZQByAGEAdABlAGQAIAB1AHMAaQBUAGcAIAB
TAHkAbgBjAGYAdQBzAGkAbwBuACAATQBlAHQAcgBvACAAUwB0AHUAZABpAG8AdwB3AHcALgBzAHk
AbgBjAGYAdQBzAGkAbwBuAC4AYwBvAG0AAAAAAGAAAAAAAKAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAASAQIBAwEEAQUBBgEHAQgBCQEKAQsBDAENAQ4BDwEQAREBEgETAANCTVAEWExTWANBVkkDRE9
DA0VYRQNNUDQDwklQA1RBUgRET0NYA0pQRwNQTKcDR01GA1BERgNUWFQES1BFRwNYTFMDTVaZAAA
=) format('truetype');
font-weight: normal;
font-style: normal;
}
[class^="sf-icon-"], [class*=" sf-icon-"] {
font-family: 'FileType_Font' !important;
speak: none;
font-size: 35px;
font-style: normal;
font-weight: normal;
font-variant: normal;
text-transform: none;
line-height: 1;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
position: relative;
top: 8px;
}
```

Output be like the below.



### Custom template

You can design the own template by preventing the default file list including buttons. The [showFileList](#) property is used to display whether the default file list or own file list when you use custom template to upload or remove the files, pass the custom UI argument as true to call `upload/remove` public method as follows:

- `UploaderObj.upload(filesData, true);`
- `UploaderObj.remove(filesData, true);`

Refer to the following code sample.

### CSHTML

```
<div class="col-lg-8 control-section">
 <div class="control_wrapper">
 <div id='dropArea'>
 Drop files here or <a href=""
id='browse'><u>Browse</u>

@Html.EJS().Uploader("UploadFiles").Selected("onFileSelect").Progress("onFileUpload").Success("onUploadSuccess").Failure("onUploadFailed").AutoUpload(false).AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
 </div>
 <div style="margin-left: 50px; padding-top:25px;">
 @Html.EJS().Button("clearbtn").Content("Clear All").Render()
 </div>
 </div>
</div>
<script>
 var dropElement = document.getElementsByClassName('control-fluid')[0];
 var filesDetails = [];
 document.getElementById('browse').onclick = function () {
 document.getElementsByClassName('e-file-select-wrap')[0].querySelector('button').click();
 return false;
 };
 document.getElementById('clearbtn').onclick = function () {
 var uploadObj = document.getElementById("UploadFiles")
 uploadObj.ej2_instances[0].element.value = '';

ej.base.detach(document.getElementById('dropArea').querySelector('.upload-list-root'));
 uploadObj.ej2_instances[0].filesData = [];
 uploadObj.ej2_instances[0].fileList = [];
 };
 var parentElement; var proxy; var progressBarContainer;
 function onFileSelect(args) {
 if
(ej.base.isNullOrUndefined(document.getElementById('dropArea').querySelector('.upload-list-root'))) {
 parentElement = ej.base.createElement('div', { className:
'upload-list-root' });

```

```

 parentElement.appendChild(ej.base.createElement('ul', {
className: 'ul-element' }));
 document.getElementById('dropArea').appendChild(parentElement);
 }
 for (var i = 0; i < args.filesData.length; i++) {
 formSelectedData(args.filesData[i], this); // create the LI
element for each file Data
 }
 this.filesData = this.filesData.concat(args.filesData);
 this.upload(args.filesData, true);
 args.cancel = true;
}
function formSelectedData(selectedFiles, proxy) {
 var liEle = ej.base.createElement('li', { className: 'file-lists',
attrs: { 'data-file-name': selectedFiles.name } });
 liEle.appendChild(ej.base.createElement('span', { className: 'file-
name ', innerHTML: selectedFiles.name }));
 liEle.appendChild(ej.base.createElement('span', { className: 'file-
size ', innerHTML: proxy.bytesToSize(selectedFiles.size) }));
 if (selectedFiles.status ===
proxy.localizedTexts('readyToUploadMessage')) {
 progressBarContainer = ej.base.createElement('span', {
className: 'progress-bar-container' });
 progressBarContainer.appendChild(ej.base.createElement('progress', {
className: 'progress', attrs: { value: '0', max: '100' } }));
 liEle.appendChild(progressBarContainer);
 }
 else {
 liEle.querySelector('.file-name').classList.add('upload-fails');
 }
 var closeIconContainer = ej.base.createElement('span', { className:
'e-icons close-icon-container' });
 ej.base.EventHandler.add(closeIconContainer, 'click', removeFiles,
proxy);
 liEle.appendChild(closeIconContainer);
 document.querySelector('.ul-element').appendChild(liEle);
 proxy.fileList.push(liEle);
}
function onFileUpload(args) {
 var li = document.getElementById('dropArea').querySelector('[data-
file-name="' + args.file.name + '"]');
 ej.base.EventHandler.remove(li.querySelector('.close-icon-
container'), 'click', removeFiles);
 var progressValue = Math.round((args.e.loaded / args.e.total) *
100);
 if (!isNaN(progressValue)) {
 li.getElementsByTagName('progress')[0].value = progressValue;
// Updating the progress bar value
 }
}
function onUploadSuccess(args) {
 var _this = this;
 var spinnerElement = document.getElementById('dropArea');
 var li = document.getElementById('dropArea').querySelector('[data-
file-name="' + args.file.name + '"]');

```

```

 if (!ej.base.isNullOrUndefined(li.querySelector('.progress-bar-
container')))) {
 ej.base.detach(li.querySelector('.progress-bar-container'));
 }
 if (args.operation === 'upload') {
 li.querySelector('.file-name').classList.add('upload-success');
 li.querySelector('.close-icon-container').classList.add('delete-
icon');
 (li.querySelector('.close-icon-container')).onclick = function
 () {
 generateSpinner(_this.uploadWrapper);
 };
 li.querySelector('.close-icon-container').onkeydown = function
 (e) {
 if (e.keyCode === 13) {
 generateSpinner(e.target.closest('.e-upload'));
 }
 };
 }
 if (args.operation === 'remove') {
 this.fileList.splice(this.fileList.indexOf(li), 1);
 this.filesData.splice(this.fileList.indexOf(li), 1);
 ej.base.detach(li);
 ej.popups.hideSpinner(spinnerElement);
 ej.base.detach(spinnerElement.querySelector('.e-spinner-pane'));
 }
 ej.base.EventHandler.add(li.querySelector('.close-icon-container'),
 'click', removeFiles, this);
 }
 function generateSpinner(targetElement) {
 ej.popups.createSpinner({ target: targetElement, width: '25px' });
 ej.popups.showSpinner(targetElement);
 }
 function onUploadFailed(args) {
 var li = document.getElementById('dropArea').querySelector('[data-
file-name="' + args.file.name + '"');
 ej.base.EventHandler.add(li.querySelector('.close-icon-container'),
 'click', removeFiles, this);
 li.querySelector('.file-name').classList.add('upload-fails');
 if (args.operation === 'upload') {
 ej.base.detach(li.querySelector('.progress-bar-container'));
 }
 }
 function removeFiles(args) {
 var status =
this.filesData[this.fileList.indexOf(args.currentTarget.parentElement)].stat
us;
 if (status === this.localizedTexts('uploadSuccessMessage')) {

this.remove(this.filesData[this.fileList.indexOf(args.currentTarget.parentEl
ement)]);

this.filesData.splice(this.fileList.indexOf(args.currentTarget.parentElement
), 1);
 }
 else {
 ej.base.detach(args.currentTarget.parentElement);
 }
 }

```



```
}
}
</script>
```

## INDEX.CSS

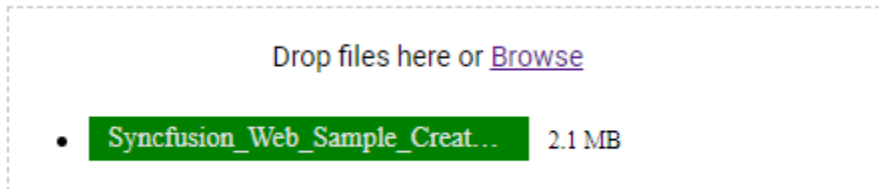
```
#dropArea {
min-height: 50px;
padding-top: 15px;
position: relative;
}
#drop {
padding: 3% 30% 3%;
display: inherit;
border: 1px dashed #c3c3cc
}
.droparea {
font-size: 13px;
}
.e-bigger .droparea {
font-size: 14px;
}
.control_wrapper {
max-width: 400px;
margin: auto;
}
.e-file-select-wrap {
display: none;
}
.e-upload {
float: none;
border: none;
}
.ul-element {
list-style: none;
width: 100%;
padding-left: 0;
}
.file-name {
padding: 8px 6px 8px 0;
font-size: 13px;
width: 46%;
display: inline-block;
position: relative;
top: 4px;
}
.e-bigger .file-name {
font-size: 14px;
}
.file-size {
padding: 4px;
font-size: 13px;
width: 18%;
display: inline-block;
position: relative;
}
```

```
.e-bigger .file-size {
font-size: 14px;
}
.file-lists {
border: 1px solid lightgray;
padding: 0 6px 0 14px;
margin-top: 15px;
position: relative;
background: #d0cdcd;
}
.file-size, .file-name {
font-family: "Helvetica Neue", "Helvetica", "Arial", "sans-serif";
text-overflow: ellipsis;
overflow: hidden;
white-space: nowrap;
}
.progress-bar-container {
display: block;
float: right;
height: 20px;
right: 13%;
top: 14px;
position: relative;
width: 20%;
}
.progress{
width: 100%;
height: 15px;
-webkit-appearance: none;
-moz-appearance: none;
}
.close-icon-container
{
cursor: pointer;
font-size: 11px;
height: 24px;
margin: 0 12px 0 22px;
padding: 0;
position: absolute;
right: 0;
width: 24px;
top: 6px;
}
.close-icon-container::before {
left: 7px;
position: inherit;
top: 7px;
content: '\e932';
}
.delete-icon::before {
content: '\e94a';
}
.close-icon-container:hover {
background-color: #d0cdcd;
border-color: transparent;
border-radius: 50%;
box-shadow: 0 0 0 transparent;
```

```
}
.highcontrast .close-icon-container:hover {
background-color: #ffd939;
}
.highcontrast .close-icon-container {
color: #ffffff;
}
.upload-success {
color: #2bc700;
}
.upload-fails {
color: #f44336;
}
progress::-webkit-progress-bar {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
}
#dropArea progress {
border: 1px solid lightgrey;
background-color: #ffffff;
border-radius: 2px;
}
.highcontrast #dropArea progress {
border: none;
background-color: #000000;
}
.highcontrast progress::-webkit-progress-bar {
border: none;
background-color: #000000;
}
.material progress::-webkit-progress-value {
border-radius: 2px;
background-color: #ff4081;
}
.bootstrap progress::-webkit-progress-value {
border-radius: 2px;
background-color: #1f496e;
}
.fabric progress::-webkit-progress-value {
background-color: #1763ff;
border-radius: 2px;
top: -66px;
}
.highcontrast progress::-webkit-progress-value {
background-color: #ffd939;
border-radius: 2px;
}
.material progress::-moz-progress-bar {
border-radius: 2px;
background-color: #ff4081;
}
.bootstrap progress::-moz-progress-bar {
border-radius: 2px;
background-color: #1f496e;
}
.fabric progress::-moz-progress-bar {
```

```
background-color: #1763ff;
border-radius: 2px;
top: -66px;
}
.highcontrast progress::-moz-progress-bar {
background-color: #ffd939;
border-radius: 2px;
}
```

Output be like the below.



See Also

- [Customize progress bar](#)
- [Customize button with HTML element](#)
- [Customize drop area](#)

## Localization in Uploader Control

The Localization library allows you to localize static text content of the uploader. The static text contains default text content of action buttons, file status, clear icon title, tooltips, and text content of drag area. Define the locale object for a culture and assign it to L10n load method.

The following are the list of keys and its values used in the uploader control:

| Key                   | Description                                                                            |
|-----------------------|----------------------------------------------------------------------------------------|
| -----                 | -----                                                                                  |
| Browse                | To customize the browse button text.                                                   |
| Clear                 | To customize the clear button text.                                                    |
| Upload                | To customize the upload button text.                                                   |
| dropFilesHint         | To customize the drop area text.                                                       |
| uploadFailedMessage   | To customize the status text when the file is failed to upload.                        |
| uploadSuccessMessage  | To customize the status text when the file is uploaded successfully.                   |
| removedSuccessMessage | To customize the status text when the file is removed the successfully from the serve. |
| removedFailedMessage  | To customize the status text while the file is failed to remove.                       |
| inProgress            | To customize the status text while the upload is in progress.                          |
| pauseUpload           | To customize the status text while the uploading is paused.                            |

- | fileUploadCancel | To customize the status text when uploading is cancelled. |
- | readyToUploadMessage | To customize the status text when the file is selected and ready to upload. |
- | invalidMaxFileSize | To customize the status text when the file size is greater than the maximum file size. |
- | invalidFileType | To customize the status text when the file type is invalid. |
- | invalidMinFileSize | To customize the status text when the file size is less than the minimum file size. |
- | remove | To customize tooltip text for remove icon. |
- | cancel | To customize tooltip text for cancel icon. |
- | delete | To customize tooltip text for delete icon. |
- | totalFiles | To customize tooltip text for total files. |
- | size | To customize tooltip text for size. |

### **CSHTML**

```
@Html.EJS().Uploader("UploadFiles").Locale("fr-CH").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
<script>
 ej.base.L10n.load({
 "fr-CH": {
 "uploader": {
 "invalidMinFileSize": "La taille du fichier est trop petite!
S'il vous plaît télécharger des fichiers avec une taille minimale de 10 Ko",
 "invalidMaxFileSize": "La taille du fichier dépasse 28 Mo",
 "invalidFileType": "Le type de fichier n'est pas autorisé",
 "Browse": "Feuilleter",
 "Clear": "Clair",
 "Upload": "Télécharger",
 "dropFilesHint": "ou Déposer des fichiers ici",
 "uploadFailedMessage": "Impossible d'importer le fichier",
 "uploadSuccessMessage": "Fichier téléchargé avec succès",
 "removedSuccessMessage": "Fichier supprimé avec succès",
 "removedFailedMessage": "Le fichier n'a pas pu être
supprimé",
 "inProgress": "Téléchargement",
 "readyToUploadMessage": "Prêt à télécharger",
 "remove": "Retirer",
 "cancel": "Annuler",
 "delete": "Supprimer le fichier",
 "totalFiles": "Total des fichiers",
 "size": "taille"
 }
 }
 })
</script>
```

### **LOCALIZATION.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```


Output be like the below.



## Accessibility

The Uploader component followed the accessibility guidelines and standards, including [ADA, Section 508](#), [WCAG 2.2](#) standards, and [WCAG roles](#) that are commonly used to evaluate accessibility.

The accessibility compliance for the Uploader component is outlined below.

|                                       |                                                                                                  |
|---------------------------------------|--------------------------------------------------------------------------------------------------|
| Accessibility Criteria                | Compatibility                                                                                    |
| --                                    | --                                                                                               |
| <a href="#">WCAG 2.2 Support</a>      |  alt="Yes" > |
| <a href="#">Section 508 Support</a>   |  alt="Yes" > |
| <a href="#">Screen Reader Support</a> |  alt="Yes" > |
| <a href="#">Right-To-Left Support</a> |  alt="Yes" > |

| [Color Contrast](#) |  |

| [Mobile Device Support](#) |  |

| [Keyboard Navigation Support](#) |  |

| [Accessibility Checker Validation](#) |  |

| [Axe-core Accessibility Validation](#) |  |

<style>

.post .post-content img {

display: inline-block;

margin: 0.5em 0;

}

</style>

<div> - All features of the component meet the requirement.</div>

<div> - Some features of the component do not meet the requirement.</div>

<div> - The component does not meet the requirement.</div>

### Keyboard interaction

The uploader control characterized with complete ARIA accessibility support that helps to be accessible by on-screen readers and other assistive technology devices.

The following are the standard keys that works on uploader control:

| **Keyboard shortcuts** | **Actions** |

| --- | --- |

| **Tab** | Moves focus to next element. |

| **Shift + Tab** | Moves focus to previous element. |

| **Enter** | Triggers corresponding action to the button element. |

| **Esc** | Closes the file browser dialog alone and cancels the upload on drop the file. |

### CSHTML

```
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

**KEYBOARD.CS**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 }
}
```

### Ensuring accessibility

The Uploader component's accessibility levels are ensured through an [accessibility-checker](#) and [axe-core](#) software tools during automated testing.

The accessibility compliance of the Uploader component is shown in the following sample. Open the [sample](#) in a new window to evaluate the accessibility of the Uploader component with accessibility tools.

See also

- [Accessibility in Syncfusion ASP.NET MVC components](#)

### Style and appearance in Uploader Component

The following content provides the exact CSS structure that can be used to modify the control's appearance based on the user preference.

#### Customizing the appearance of File Upload wrapper element

Use the following CSS to customize the appearance of wrapper element.

`css

*/ To specify height /*



```
.e-upload.e-control-wrapper, .e-bigger.e-small .e-upload.e-control-wrapper {
height: 300px;
width: 300px;
}
`
```

#### Customizing the File Upload browse button

Use the following CSS to customize the File Upload browse button

```
`css

/ To specify font size and color /

.e-upload .e-file-select-wrap .e-btn, .e-upload .e-upload-actions .e-btn, .e-bigger.e-small .e-upload .e-
file-select-wrap .e-btn, .e-bigger.e-small .e-upload .e-upload-actions .e-btn {
font-family: cursive;
height: 40px;
background-color: aquamarine;
color: coral;
}
`
```

#### Customizing the File Upload content

Use the following CSS to customize the File Upload content

```
`css

/ To specify font size and color /

.e-upload .e-file-select-wrap .e-file-drop, .e-bigger.e-small .e-upload .e-file-select-wrap .e-file-drop {
font-size: 20px;
color: aqua;
}
`
```

#### Customizing the uploaded file container in File Upload

Use the following CSS to customize the uploaded file container in File Upload

```
`css

/ To specify background color /

.e-upload .e-upload-files .e-upload-file-list {
background-color: beige;
}
`
```

## See Also

- [Customize the appearance of uploader using a template](#)

## How To

### Hide default drop area

You can achieve this behavior by overriding the corresponding uploader styles. Override the following styles to hide the default drop area behavior.

- .e-upload.e-control
- .e-upload .e-file-select
- .e-upload .e-file-drop

## CSHTML

```
@Html.EJS().Uploader("UploadFiles").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
```

## INDEX.CSS

```
#droparea {
padding: 50px 25px;
margin: 30px auto;
border: 1px solid #c3c3c3;
text-align: center;
width: 20%;
display: inline-flex;
}
.e-file-select,
.e-file-drop {
display: none;
}
body .e-upload-drag-hover {
outline: 2px dashed brown;
}
#uploadfile {
width: 60%;
display: inline-flex;
margin-left: 5%;
}
.e-control .e-file-select,
.e-control .e-file-drop {
display: none;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Preview images before uploading

The uploader control allows to create preview images before uploading. The preview images can be created by reading the file using [selected](#) event. Also, the user can create preview images after uploading to server using [success](#) event. Refer to the following link to learn about how to create image preview.

### [Image Preview in MVC](#)

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Achieve invisible upload

You can achieve the invisible upload feature by using the [selected](#) event in uploader control.

Refer to the following example.

### CSHTML

```
<div class="control_wrapper">
 <div id='preview'></div>
 @Html.EJS().Uploader("UploadFiles").AllowedExtensions(".png, .jpg,
 .jpeg").Selected("onSelect").AsyncSettings(new
 Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
 "https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
 "https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
 function onSelect(args) {
 for (var i = 0; i < args.filesData.length; i++) {
 var liparentDiv = ej.base.createElement('div', { className:
 'image-list' });
 var liImage = ej.base.createElement('img', { className: 'image'
 });

 liparentDiv.appendChild(liImage);
 readURL(liImage, args.filesData[i]);
 document.getElementById('preview').appendChild(liparentDiv);
 }
 args.cancel = true;
}
function readURL(liImage, file) {
 var imgPreview = liImage;
 var imageFile = file.rawFile;
 var reader = new FileReader();
 reader.addEventListener('load', () => {
 imgPreview.src = reader.result;
 }, false);
 if (imageFile) {
 reader.readAsDataURL(imageFile);
 }
}
</script>
```

### INDEX.CSS

```
.control_wrapper {
```

```

 max-width: 505px;
 margin: auto;
 }
 #preview {
 border: 2px dashed #ddd;
 padding: 15px;
 }
 .image-list {
 width: 134px;
 height: 117px;
 margin-right: 4px;
 border: 1px solid lightgrey;
 display: inline-block;
 }
 .image {
 width: 134px;
 height: 117px;
 margin-right: 4px;
 display: inline-block;
 }
 .e-control ul {
 border: 1px solid #ddd;
 }
 .e-control .e-file-select,
 .e-control .e-file-drop {
 display: none;
 }
 .e-control {
 margin-top: 10px;
 }
}

```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Customize progressbar

You can customize the progress bar's size, color, and background by overriding the styles in uploader control. Refer to the following example.

### CSHTML

```

<div class="custom_progress">
 @Html.EJS().Uploader("UploadFiles").AsyncSettings(new
 Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
 "https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
 "https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>

```

### INDEX.CSS

```

.custom_progress {
 max-width: 400px;
 margin: 0 auto;
}

```

```
.custom_progress .e-upload .e-upload-files .e-upload-file-list .e-file-
container .e-progress-inner-wrap .e-upload-progress-bar.e-upload-progress {
 background: yellow;
 height: 4px;
}
.custom_progress .e-upload .e-upload-files .e-upload-file-list .e-file-
container .e-progress-inner-wrap {
 height: 4px;
 background-color: lightblue;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Sort the selected files

You can sort the selected files in uploader control by using the [selected](#) event. Refer to the following example.

### CSHTML

```
<div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").Selected("onSelect").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
 var initial = true
 function onSelect(args) {
 if (initial) { initial = false; return; }
 args.isModified = true;
 var uploadObj =
document.getElementById("UploadFiles").ej2_instances[0];
 let oldFiles = uploadObj.GetFilesData();
 let filesData = args.filesData.concat(oldFiles);
 let modifiedData = sortFileList(filesData);
 args.modifiedFilesData = modifiedData;
 }
 function sortFileList(filesData) {
 let files = filesData;
 let fileNames = [];
 for (let i = 0; i < files.length; i++) {
 fileNames.push(files[i].name);
 }
 let sortedFileNames = fileNames.sort();
 let sortedFilesData = [];
 let index = 0;
 for (let name of sortedFileNames) {
 for (let i = 0; i < files.length; i++) {
 if (name === files[i].name) {
 sortedFilesData.push(files[i]);
 }
 }
 }
 }
}
```

```

 return sortedFilesData;
 }
</script>

```

### INDEX.CSS

```

.control_wrapper {
 max-width: 400px;
 margin: 0 auto;
}

```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Get the total size of selected files

You can get the total size of selected files before uploading it to the designated server. This can be achieved by using the [selected](#) event. Refer to the following example to calculate the total file size.

### CSHTML

```

<div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").Selected("onSelect").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
 function onSelect(args) {
 var uploadObj =
document.getElementById("UploadFiles").ej2_instances[0];
 var totalSize = 0;
 for (var i = 0; i < args.filesData.length; i++) {
 var file = args.filesData[i];
 totalSize = totalSize + file.size;
 }
 var size = uploadObj.bytesToSize(totalSize);
 alert("Total select file's size is " + size);
 }
</script>

```

### INDEX.CSS

```

.control_wrapper {
 max-width: 400px;
 margin: 0 auto;
}

```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

### Customize button with HTML element

The uploader control allows you to customize the action buttons by using [buttons](#) property. Refer to the following example.

#### CSHTML

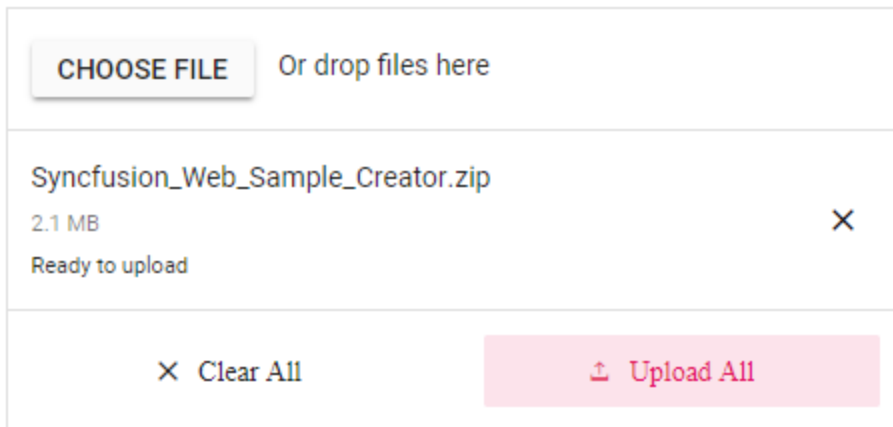
```
<div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").AutoUpload("false").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
 var uploadEle = ej.base.createElement('span', { className: 'upload e-
icons' });
 uploadEle.innerHTML = 'Upload All';
 var clearEle = ej.base.createElement('span', { className: 'remove e-
icons' });
 clearEle.innerHTML = 'Clear All';
 window.onload = function (args) {
 var uploaderObj =
document.getElementById("UploadFiles").ej2_instances[0];
 uploaderObj.setProperties({
 buttons: {
 browse: 'Choose file',
 clear: clearEle,
 upload: uploadEle
 }
 })
 }
</script>
```

#### INDEX.CSS

```
.control_wrapper {
 max-width: 400px;
 margin: 0 auto;
}
.e-upload .e-upload-actions .e-file-clear-btn,
.e-upload .e-upload-actions .e-file-upload-btn {
 -webkit-tap-highlight-color: transparent;
 background-color: #ddd;
 border-color: #c3c3c3;
 color: #000000;
 box-shadow: none;
 width: 50%;
 margin: 0;
 text-transform: none;
 padding: 8px 0;
}
.e-upload .e-upload-actions {
 width: 100%;
}
```

Output be like the below.



**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

Add confirm dialog to remove the files

You can customize the uploader control using confirm dialog before removing the files.

Here, ej2 dialog is used as confirm dialog. Refer to the following example.

#### CSHTML

```
<div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").Removing("onremove").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()

@Html.EJS().Dialog("dialog").Target("body").Visible(false).Width("250px").Co
ntent("Confirm to remove the file?").Buttons(ViewBag.confirmbutton).Render()
</div>
<script>
var removeFile = [], dialog;
window.onload = function () {
 dialog = document.getElementById("dialog").ej2_instances[0];
}
function onremove(args) {
 removeFile = [];
 args.cancel = true;
 removeFile.push(args.filesData);
 dialog.show();
}
function onClick(args) {
 var uploadObj =
document.getElementById("UploadFiles").ej2_instances[0];
 dialog.hide();
 uploadObj.remove(removeFile[0], false, true);
}
function Close() {
 dialog.hide();
}
```



```
}
</script>
```

### INDEX.CSS

```
.control_wrapper {
 max-width: 400px;
 margin: 0 auto;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Add additional data on upload

The uploader control allows to add additional data on file upload, which is used to get in the server-side.

By using [uploading](#) event and its customFormData argument, this behavior can be achieved.

### CSHTML

```
<div class="control_wrapper">

@Html.EJS().Uploader("fileupload").Uploading("onFileUpload").AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
function onFileUpload(args) {
 args.customFormData = [{ 'name': 'Syncfusion INC' }];
}
</script>
```

### INDEX.CSS

```
.control_wrapper {
 max-width: 505px;
 margin: auto;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Validate image/\* on drop

The uploader control allows you to select all types of images using the *image/\** to [allowedExtensions](#) property. You can directly set it to accept the attribute of uploader element.

By default, it is working fine when you select a file by clicking the browse button. But, this behavior is not supported to drag and drop the files for selection.

### CSHTML

```

<div class="control_wrapper">

@Html.EJS().Uploader("fileupload").AutoUpload(false).AllowedExtensions("image/*").Selected("onSelect").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
function onSelect(args) {
 var uploadObj = document.getElementById("fileupload").ej2_instances[0];
 if (event.type === 'change') {
 var allImages = ['jpg', 'jpeg', 'gif', 'tiff', 'bpg'];
 var files = args.filesData;
 var modifiedFiles = [];
 for (var i = 0; i < files.length; i++) {
 var file = files[i];
 if (allImages.indexOf(file.type) === -1) {
 file.status = 'File type is not allowed';
 file.statusCode = '0';
 }
 modifiedFiles.push(file);
 }
 args.isModified = true;
 args.modifiedFilesData = modifiedFiles.concat(uploadObj.filesData);
 }
}
</script>

```

### INDEX.CSS

```

.control_wrapper {
 max-width: 505px;
 margin: auto;
}

```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

Determine whether uploader has file input (required validation)

By setting **required** attribute to uploader input element, you can validate the file input has any value in it.

In the below sample, set required attribute to the uploader input element and showcase the validation failure message using `data-required-message` attribute.

### CSHTML

```

<div class="col-lg-8 control-section">
 <div class="control_wrapper">
 <div class="col-lg-12 control-section">
 <h4 class="form-title">Photo Contest</h4>
 <div class="control_wrapper" id="control_wrapper">
 <form id="form1" method="post">

```

```

 <div class="form-group" style="padding-top: 40px; float:
left">

 <div class="e-float-input">
 <input type="text" id="name" name="name" data-
required-message="* Enter your name" required="" data-msg-
containerid="nameError">

 <label class="e-float-text e-label-top"
for="name">Name</label>
 </div>
 <div id="nameError"></div>
 </div>
 <div id="dropArea">
 <div id="uploadError" style='float: right;'></div>
 <div id='customBrowse' class="form-group
dropUpload">

 Drop image here...

@Html.EJS().Uploader("UploadFiles").DropArea(".dropUpload").Selected("onFile
Select").AllowedExtensions("image/*").Multiple(true).AutoUpload(false).Async
Settings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
 </div>
 </div>
 <div class="submitBtn">
 <button type="button" class="submit-btn e-btn"
id="submit-btn">Submit</button>
 <div class="desc">*This button is not a submit
type and the form submit handled from externally.</div>
 </div>
</form>

@Html.EJS().Dialog("confirmationDialog").Buttons(ViewBag.DefaultButtons).Wid
th("335px").Visible(false).Content("Your details has been updated
successfully, Thank
you").Target("#control_wrapper").IsModal(true).AnimationSettings(new
Syncfusion.EJ2.Popups.DialogAnimationSettings() { Effect =
Syncfusion.EJ2.Popups.DialogEffect.Zoom }).Render()
 </div>
</div>
</div>
<script>
 var formID, confirm;
 window.onload = function () {
 inputElement = document.getElementById('upload');
 formID = document.getElementById('form1');
 confirm =
document.getElementById("confirmationDialog").ej2_instances[0];
 formObj = new ej.inputs.FormValidator(formID, options);
 document.getElementById('customBrowse').onclick = () => {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
 };
 document.getElementById('submit-btn').onclick = () => {
 onFormSubmit();

```

```

 };
}
var options = {
 customPlacement: function (inputElement, errorElement) {
 inputElement = inputElement.closest('.form-
group').querySelector('.error');
 inputElement.parentElement.appendChild(errorElement);
 },
 rules: {
 'name': {
 required: true
 }
 }
};
function onFileSelect(args) {
 if (args.filesData.length > 0) {
 if (document.getElementsByClassName('upload-image').length > 0)
 {
 detach(document.getElementsByClassName('imgWrapper')[0]);
 }
 var imageTag = ej.base.createElement('IMG', { className:
'upload-image', attrs: { 'alt': 'Image' } });
 var wrapper = ej.base.createElement('span', { className:
'imgWrapper' });
 wrapper.appendChild(imageTag);
 var rootFile = document.getElementsByClassName('dropUpload')[0];
 rootFile.insertBefore(wrapper, rootFile.firstChild);
 readURL(wrapper, args.filesData[0]);
 }
 args.cancel = true;
}
function readURL(li, args) {
 var preview = li.querySelector('.upload-image');
 var file = args.rawFile;
 var reader = new FileReader();
 reader.addEventListener('load', () => { preview.src = reader.result;
}, false);
 if (file) { reader.readAsDataURL(file); }
}
var options = {};
formID = document.getElementById('form1');
var formObj = new ej.inputs.FormValidator(formID, options);
function onFormSubmit() {
 var formStatus = formObj.validate();
 if (formStatus) {
 formObj.element.reset();
 }
 ej.base.detach(document.getElementsByClassName('imgWrapper')[0]);
 confirm.show();
}
document.querySelector('#UploadFiles').setAttribute('data-required-
message', '* Choose your image to upload');
document.querySelector('#UploadFiles').setAttribute('required', '');
document.querySelector('#UploadFiles').setAttribute('data-msg-
containerid', 'uploadError');
}
function dlgButtonClick() {

```

```

 confirm.hide();
 }
</script>

```

## INDEX.CSS

```

.control_wrapper {
 max-width: 400px;
 margin: auto;
}
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}
.address-field {
 resize: none;
}
#dropArea .dropUpload .upload-image {
 height: 140px;
 width: 140px;
}
#dropArea .dropUpload {
 float: right;
 text-align: center;
 vertical-align: middle;
 line-height: 12;
 overflow: hidden;
 border: 1px dashed;
 width: 150px;
 height: 150px;
}
.e-upload {
 visibility: hidden;
}
#control_wrapper {
 max-width: 500px;
 margin: auto;
 border: 0.5px solid #BEBEBE;
 box-shadow: 0 1px 3px 0 rgba(0, 0, 0, 0.36);
 padding: 1% 4% 7%;
 background: #f9f9f9;
}
.e-error {
 padding-top: 3px;
}
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}

.submit-btn {
 margin-top: 15px;
 margin: auto 100px;
}
.submitBtn .desc {
 margin: 2% 23% 0 18%;
}
.submitBtn {

```

```

 text-align: center;
}
.form-support {
 width: 100%;
}
.success .form-support {
 display: none;
}
.success .successmsg {
 border: 0.5px solid green;
 padding: 10%;
 color: green;
}
#form1 {
 position: relative;
 top: 14%;
}
.form-support td {
 width: 100%;
 padding-top: 4%;
}
.e-upload {
 float: none;
}
.choose-file {
 width: 60%;
}
#browse {
 float: right;
 margin-right: -113px;
 margin-top: -27px;
}
.form-title {
 text-align: center;
}

```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Achieve file upload programmatically

You can upload a file programmatically using `upload` method.

The selected files data, get from `getFilesData` public method in uploader.

The upload method behaves differently based on its arguments.

- If this method receives any files as arguments, those files only start to upload.
- If it has no argument then all the selected files are will start to upload.

#### CSHTML

```
<div class="col-lg-8 control-section">
```

```

<div class="control_wrapper">
@Html.EJS().Uploader("UploadFiles").AutoUpload(false).AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>

<button id='first' class='e-btn e-control'>Upload 0th File</button>

<button id='full' class='e-btn e-control'>Upload All Files</button>

</div>
<script>
window.onload = function () {
 var uploadObj =
document.getElementById("UploadFiles").ej2_instances[0];
 document.getElementById('first').onclick = (args) => {
 uploadObj.upload(uploadObj.getFilesData()[0]);
 };
 document.getElementById('full').onclick = (args) => {
 uploadObj.upload(uploadObj.getFilesData());
 };
}
</script>

```

## INDEX.CSS

```

.control_wrapper {
 max-width: 500px;
 margin: auto;
}
.e-upload {
 width: 100%;
 position: relative;
 margin-top: 15px;
 margin-bottom: 15px;
}
.e-upload-actions {
 display: none;
}
.e-btn {
 text-transform: none;
}
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}

```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

Check file size before uploading it

By using [uploading](#) event, you can get the file size before upload it to server.

File object contains the file size in bytes only.

You can convert the size to standard formats (KB or MB) using `bytesToSize` method.

### CSHTML

```
@using Syncfusion.EJ2
<div class="col-lg-8 control-section">
 <div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").Uploading("onBeforeUpload").AutoUpload(f
alse).AsyncSettings(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings {
SaveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Save",
RemoveUrl = "https://ej2.syncfusion.com/services/api/uploadbox/Remove"
}).Render()
 </div>
</div>
<script>
 function onBeforeUpload(args) {
 var uploadObj =
document.getElementById("UploadFiles").ej2_instances[0];
 // get the file size in bytes
 var sizeInBytes = args.fileData.size;
 // get the file size in standard format
 alert("File size is: " + uploadObj.bytesToSize(sizeInBytes))
 }
</script>
```

### INDEX.CSS

```
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

Check the MIME type of file before upload it

By using [uploading](#) event, you can get the file MIME type before uploading it to server.

In the below sample, file MIME type is shown in the alert box before file start to upload.

### CSHTML

```
<div class="col-lg-8 control-section">
 <div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").Uploading("onBeforeUpload").AutoUpload(f
alse).DropArea(".control-fluid").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
```



```
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
</div>
<script>
 function onBeforeUpload(args) {
 // get the file MIME type
 alert("File MIME type is: " + args.fileData.rawFile.type);
 }
</script>
```

### INDEX.CSS

```
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

Trigger click event of input file from external button

You can trigger the click event of input file from external button using `click` event of button. In the below sample, triggered click event of input file from `Essential JavaScript 2 Button`.

### CSHTML

```
<div class="col-lg-8 control-section">
 <div class="control_wrapper">
 <div id="dropArea">
 Drop image (JPG, PNG) files here or <button
class='e-btn e-control' id="browse">Browse</button>
 </div>
 </div>
</div>
<script>
 window.onload = function () {
 document.getElementById('browse').onclick = () => {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
 };
 }
</script>
```

### INDEX.CSS

```
.control_wrapper {
 max-width: 500px;
 margin: auto;
}
#dropArea {
 border: 1px dashed #c3c3cc;
 text-align: center;
 padding: 20px 0 10px;
}
.e-file-select-wrap {
 display: none;
}
.control_wrapper .e-upload .e-upload-drag-hover {
 margin: 0;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Open and edit the uploaded files

The uploader control allows you to modify the file after uploading to the server, which can be achieved using [success](#) event of the uploader.

You can retrieve the saved file path in the uploader success event and assign it to custom attribute (data-file-name) value of the respective file list element to open the uploaded file. Click the respective file element to create a new request along with saved file path using http header. In the server-side, get the file path from the header and open the file using `process.start` method.

#### CSHTML

```
@using Syncfusion.EJ2
<div class="control_wrapper">

@Html.EJS().Uploader("UploadFiles").Success("onUploadSuccess").AsyncSettings
(new Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove" }).Render()
</div>
<script>
function onUploadSuccess(args) {
 // fetching the generated li elements
 var liElements = this.uploadWrapper.querySelectorAll('.e-upload-file-
list');
 for (var i = 0; i < liElements.length; i++) {
 if (liElements[i].getAttribute('data-file-name') == args.file.name)
 {
 liElements[i].addEventListener('click', () => { openFile(args,
event) })
 // File path have to update from server end in response status
description.
 liElements[i].setAttribute('file-path',
args.e.target.statusText);
 }
 }
}
```

```

}
function openFile(args, e) {
 if (!e.target.classList.contains('e-file-delete-btn') &&
 !e.target.classList.contains('e-file-remove-btn'))
 {
 var ajax = new XMLHttpRequest();
 // create new request for open the selected file
 ajax.open("POST", '/Home/openFile');
 var liElements = document.getElementsByClassName('e-
upload')[0].querySelectorAll('.e-upload-file-list');
 for (var i = 0; i < liElements.length; i++) {
 if (liElements[i].getAttribute('data-file-name') ==
args.file.name) {
 // Added the file path in header to get it in server side.
 ajax.setRequestHeader('filePath',
liElements[i].getAttribute('file-path').toString());
 }
 }
 ajax.send();
 }
}
</script>

```

### SERVER-SIDE.CS

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
namespace EJ2CoreSampleBrowser.Controllers.TextBoxes
{
 public partial class UploaderController : Controller
 {
 public ActionResult DefaultFunctionalities()
 {
 return View();
 }
 public void Save() {
 if (!System.IO.File.Exists(fileSavePath))
 {
 httpPostedFile.SaveAs(fileSavePath);
 HttpResponseMessage Response = System.Web.HttpContext.Current.Response;
 Response.Clear();
 Response.ContentType = "application/json; charset=utf-8";
 <!-- Sending the file path to client side -->
 Response.StatusDescription = fileSavePath;
 Response.End();
 }
 }
 }
 [AcceptVerbs("Post")]
 public void openFile()
 {
 // Check whether the file is available in the corresponding location
 if
(System.IO.File.Exists(Request.Headers.GetValues("filePath").First()))

```

```
{
 // This will open the selected file from server location in desktop
 Process.Start(Request.Headers.GetValues("filePath").First());
}
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

Resize images before uploading it to the server

You can customize the dimension of the images before uploading it to the server.

By using [selected](#) event, you can get the selected file information as type of an object. From the obtained image file information, create a new canvas and render an image with the custom dimensions. Refer the corresponding code snippet as follows.

### CSHTML

```
<div id='dropArea'>
 Drop files here or <a href=""
id='browse'><u>Browse</u>

@Html.EJS().Uploader("UploadFiles").DropArea("#dropArea").Selected("onFileSe
lect").Progress("onFileUpload").AsyncSettings(new
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Save", RemoveUrl =
"https://ej2.syncfusion.com/services/api/uploadbox/Remove"
}).AllowedExtensions("image/*").Success("onUploadSuccess").Removing("onFileR
emove").Failure("onUploadFailed").Render()
</div>
<script>
 var uploadObj;
 window.onload = function () {
 uploadObj = document.getElementById("UploadFiles").ej2_instances[0];
 }
 document.getElementById('browse').onclick = function () {
 document.getElementsByClassName('e-file-select-
wrap')[0].querySelector('button').click();
 return false;
 };
 var parentElement; var proxy; var progressBarContainer;
 function onFileSelect(args) {
 args.cancel = true;
 if
(ej.base.isNullOrUndefined(document.getElementById('dropArea').querySelector
('.upload-list-root'))) {
 parentElement = ej.base.createElement('div', { className:
'upload-list-root' });
 parentElement.appendChild(ej.base.createElement('ul', {
className: 'ul-element' }));
 document.getElementById('dropArea').appendChild(parentElement);
 }
 for (var i = 0; i < args.filesData.length; i++) {
```

```

 formSelectedData(args.filesData[i], this); // create the LI
element for each file Data
 }
 this.filesData = this.filesData.concat(args.filesData);
 var file = args.filesData[0].rawFile;
 var width;
 var height;
 var img = document.createElement("img");
 var reader = new FileReader();
 reader.onload = function (e) { img.src = e.target.result; };
 reader.readAsDataURL(file);
 var imgs = new Image();
 img.onload = function () {
 width = this.width;
 height = this.height;
 onNewImg(height, width, img, args.filesData[0]);
 };
 imgs.src = img.src;
}
// to create canvas and update our custom dimensions
function onNewImg(height, width, img, file) {
 var canvas = document.createElement("canvas");
 var ctx = canvas.getContext("2d");
 ctx.drawImage(img, 0, 0);
 var MAX_WIDTH = 1000;
 var MAX_HEIGHT = 600;
 if (width > height) {
 if (width > MAX_WIDTH) {
 height *= MAX_WIDTH / width;
 width = MAX_WIDTH;
 }
 }
 else {
 if (height > MAX_HEIGHT) {
 width *= MAX_HEIGHT / height;
 height = MAX_HEIGHT;
 }
 }
 canvas.width = width;
 canvas.height = height;
 var ctx1 = canvas.getContext("2d");
 ctx1.drawImage(img, 0, 0, width, height);
 newImage = canvas.toDataURL("image/png");
 var blobBin = atob(newImage.split(',')[1]);
 var array = [];
 for (var i = 0; i < blobBin.length; i++) {
 array.push(blobBin.charCodeAt(i));
 }
 var newBlob = new Blob([new Uint8Array(array)], { type: 'image/png'
});

 var newFile = createFile(newBlob, file);
 uploadObj.upload(newFile, true);
}
// To create File object to upload
function createFile(image, file) {
 var newFile = {
 name: file.name,

```

```

 rawFile: image,
 size: image.size,
 type: file.type,
 validationMessage: '',
 statusCode: '1',
 status: 'Ready to Upload'
 };
 return newFile;
}
function formSelectedData(selectedFiles, proxy) {
 var liEle = ej.base.createElement('li', { className: 'file-lists',
 attrs: { 'data-file-name': selectedFiles.name } });
 liEle.appendChild(ej.base.createElement('span', { className: 'file-
 name ', innerHTML: selectedFiles.name }));
 liEle.appendChild(ej.base.createElement('span', { className: 'file-
 size ', innerHTML: proxy.bytesToSize(selectedFiles.size) }));
 if (selectedFiles.status ===
 proxy.localizedTexts('readyToUploadMessage')) {
 progressBarContainer = ej.base.createElement('span', {
 className: 'progress-bar-container' });
 progressBarContainer.appendChild(ej.base.createElement('progress', {
 className: 'progress', attrs: { value: '0', max: '100' } }));
 liEle.appendChild(progressBarContainer);
 }
 else {
 liEle.querySelector('.file-name').classList.add('upload-fails');
 }
 var closeIconContainer = ej.base.createElement('span', { className:
 'e-icons close-icon-container' });
 ej.base.EventHandler.add(closeIconContainer, 'click', removeFiles,
 proxy);
 liEle.appendChild(closeIconContainer);
 document.querySelector('.ul-element').appendChild(liEle);
 proxy.fileList.push(liEle);
}
function onFileUpload(args) {
 var li = document.getElementById('dropArea').querySelector('[data-
 file-name="' + args.file.name + '"]');
 ej.base.EventHandler.remove(li.querySelector('.close-icon-
 container'), 'click', removeFiles);
 var progressValue = Math.round((args.e.loaded / args.e.total) *
 100);
 if (!isNaN(progressValue)) {
 li.getElementsByTagName('progress')[0].value = progressValue;
 // Updating the progress bar value
 }
}
function onUploadSuccess(args) {
 var _this = this;
 var spinnerElement = document.getElementById('dropArea');
 var li = document.getElementById('dropArea').querySelector('[data-
 file-name="' + args.file.name + '"]');
 if (!ej.base.isNullOrUndefined(li.querySelector('.progress-bar-
 container'))) {
 ej.base.detach(li.querySelector('.progress-bar-container'));
 }
}

```

```

 if (args.operation === 'upload') {
 li.querySelector('.file-name').classList.add('upload-success');
 li.querySelector('.close-icon-container').classList.add('delete-
icon');
 (li.querySelector('.close-icon-container')).onclick = function
 () {
 generateSpinner(_this.uploadWrapper);
 };
 li.querySelector('.close-icon-container').onkeydown = function
 (e) {
 if (e.keyCode === 13) {
 generateSpinner(e.target.closest('.e-upload'));
 }
 };
 }
 if (args.operation === 'remove') {
 this.filesData.splice(this.fileList.indexOf(li), 1);
 this.fileList.splice(this.fileList.indexOf(li), 1);
 ej.base.detach(li);
 ej.popups.hideSpinner(spinnerElement);
 ej.base.detach(spinnerElement.querySelector('.e-spinner-pane'));
 }
 ej.base.EventHandler.add(li.querySelector('.close-icon-container'),
 'click', removeFiles, this);
 console.log("The selected file resized and uploaded successfully");
 }
 function generateSpinner(targetElement) {
 ej.popups.createSpinner({ target: targetElement, width: '25px' });
 ej.popups.showSpinner(targetElement);
 }
 function onUploadFailed(args) {
 var li = document.getElementById('dropArea').querySelector('[data-
file-name="' + args.file.name + '"]');
 ej.base.EventHandler.add(li.querySelector('.close-icon-container'),
 'click', removeFiles, this);
 li.querySelector('.file-name').classList.add('upload-fails');
 if (args.operation === 'upload') {
 ej.base.detach(li.querySelector('.progress-bar-container'));
 }
 }
 function removeFiles(args) {
 var status =
this.filesData[this.fileList.indexOf(args.currentTarget.parentElement)].stat
us;
 if (status === this.localizedTexts('uploadSuccessMessage')) {
 this.remove(this.filesData[this.fileList.indexOf(args.currentTarget.parentElement)], true);
 }
 else {
 ej.base.detach(args.currentTarget.parentElement);
 }
 }
 function onFileRemove(args) {
 args.postRawFile = false;
 }
</script>

```

**INDEX.CSS**

```
#dropArea {
 min-height: 50px;
 padding-top: 15px;
 position: relative;
}
#drop {
 padding: 3% 30% 3%;
 display: inherit;
 border: 1px dashed #c3c3cc
}
.droparea {
 font-size: 13px;
}
.e-file-select-wrap {
 display: none;
}
.e-upload {
 float: none;
 border: none;
}
.ul-element {
 list-style: none;
 width: 100%;
 padding-left: 0;
}
.file-name {
 padding: 8px 6px 8px 0;
 font-size: 13px;
 width: 46%;
 display: inline-block;
 position: relative;
 top: 4px;
}
.file-size {
 padding: 4px;
 font-size: 13px;
 width: 18%;
 display: inline-block;
 position: relative;
}
.file-lists {
 border: 1px solid lightgray;
 padding: 0 6px 0 14px;
 margin-top: 15px;
 position: relative;
 background: rgba(0, 0, 0, 0.04);
}
.file-size, .file-name {
 font-family: "Helvetica Neue", "Helvetica", "Arial", "sans-serif";
 text-overflow: ellipsis;
 overflow: hidden;
 white-space: nowrap;
}
```



```
span.progress-bar-container {
 display: block;
 float: right;
 height: 20px;
 right: 13%;
 top: 14px;
 position: relative;
 width: 20%;
}
.progress {
 width: 100%;
 height: 15px;
 -webkit-appearance: none;
}
.close-icon-container {
 cursor: pointer;
 font-size: 11px;
 height: 24px;
 margin: 0 12px 0 22px;
 padding: 0;
 position: absolute;
 right: 0;
 width: 24px;
 top: 6px;
}
.close-icon-container.e-icons::before {
 left: 7px;
 position: inherit;
 top: 7px;
 content: '\e932';
}
.close-icon-container.delete-icon::before {
 content: '\e94a';
}
.close-icon-container:hover {
 background-color: rgba(0, 0, 0, 0.12);
 border-color: transparent;
 border-radius: 50%;
 box-shadow: 0 0 0 transparent;
}
.highcontrast .close-icon-container:hover {
 background-color: #ffd939;
 color: black;
}
.highcontrast .close-icon-container {
 color: #ffffff;
}
.upload-success {
 color: #2bc700;
}
.upload-fails {
 color: #f44336;
}
progress::-webkit-progress-bar {
 border: 1px solid lightgrey;
 background-color: #ffffff;
 border-radius: 2px;
```

```
}
#dropArea progress {
 border: 1px solid lightgrey;
 background-color: #ffffff;
 border-radius: 2px;
}
progress::-webkit-progress-value {
 border-radius: 2px;
 background-color: #ff4081;
}
progress::-moz-progress-bar {
 border-radius: 2px;
 background-color: #ff4081;
}
#dropArea span a {
 color: #ff4081;
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

#### Convert image into binary format after uploading

By default, the file upload control saves the uploaded image files in physical directories. Also, you can convert the images into binary format at server-side before saving the uploaded images.

To retrieve binary format of image files, convert the posted file's input stream into binary reader and read as bytes using ReadBytes method.

Refer to the below server-side code snippet

```
`csharp
[[AcceptVerbs("Post")]
public IActionResult Save(IList<IFormFile> UploadFiles)
{
 IFormFile uploadedImage = UploadFiles.FirstOrDefault();
 if (uploadedImage.ContentType.ToLower().StartsWith("image/"))
 // Check whether the selected file is image
 {
 byte[] b;
 using (BinaryReader br = new BinaryReader(uploadedImage.OpenReadStream()))
 {
 b = br.ReadBytes((int)uploadedImage.OpenReadStream().Length);
 // Convert the image in to bytes
 }
 Response.StatusCode = 200;
 }
}
```

```
}
return Content("");
}
```

**Note:** You can also explore [ASP.NET MVC File Upload](#) feature tour page for its groundbreaking features. You can also explore our [ASP.NET MVC File Upload example](#) to understand how to browse the files which you want to upload to the server.

## Migration from Essential JS 1

This article describes the API migration process of File Upload component from Essential JS 1 to Essential JS 2.

### Accessibility

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Localization | **Property** : locale <br/><br/>

@Html.EJ().Uploadbox("UploadDefault")<br/>.Locale("es-ES") | **Property** : locale <br/><br/>

@Html.EJS().<br/>Uploader("UploadFiles").<br/>Locale("es-ES").Render() |

| Right to left | **Property**: enableRTL <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>EnableRTL(true) | **Property**: enableRTL <br/><br/>

@Html.EJS().Uploader("UploadFiles").<br/>EnableRtl(true).Render() |

### File list

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

|-----|-----|-----|

| Show/Hide the selected files | **Property** : showFileDetails <br/><br/>

@Html.EJ().Uploadbox("UploadDefault")<br/>.ShowFileDetails(false) | **Property** : showFileList

<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>ShowFileList(false).Render() |

| Customizing the file list | Not Applicable | **Property** : template <br/><br/>

@Html.EJS().Uploader("UploadFiles").<br/>Template("#Templateid").Render() |

| Get the files in sorted form | Not Applicable | **Method**: SortFileList<br/><br/>

@Html.EJS().Uploader("UploadFiles")<br/>.Render()<br/><br/>var uploadobj =

document.<br/>getElementById("UploadFiles").<br/>ej2\_instances[0];

<br/>uploadObj.sortFileList(files); <br/> |

| Clearing File List | Not Applicable | **Method**: ClearAll <br/><br/>

@Html.EJS().Uploader("UploadFiles").<br/>Render()<br/><br/>var uploadobj =

document.<br/>getElementById("UploadFiles").<br/>ej2\_instances[0];

<br/>uploadObj.clearAll();<br/>} <br/> |

| Event triggers when clearing Files | Not Applicable | **Event** : clearing <br/><br/>

@Html.EJS().Uploader("UploadFiles").<br/>Clearing("onClearing").Render()<br/><br/> function

onClearing(ClearingEventArgs): void { } <br/> |

## File selection

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Select multiple files to upload | **Property** : multipleFilesSelection <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>MultipleFilesSelection(true) | **Property** : multiple  
<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>Multiple(true).Render() |

| Set minimum file size to upload | **Not Applicable** | **Property** : minFileSize <br /><br />

@Html.EJS().Uploader("UploadFiles").<br/>MinFileSize(1024).Render() |

| Set maximum file size to upload | **Property** : fileSize <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>FileSize(5000) | **Property** : maxFileSize <br/><br/>

@Html.EJS().Uploader("UploadFiles").<br/>MaxFileSize(5000).Render() |

| Allowed file types to select | **Property** : extensionsAllow <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>ExtensionsAllow(".zip") | **Property**:  
allowedExtensions <br/><br/>

@Html.EJS().Uploader("UploadFiles").<br/>AllowedExtensions(".pdf").Render() |

| Restricted files types to select | **Property**: extensionsDeny <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>ExtensionsDeny(".docx") | **Not Applicable** |

| Display only selected details in File list | **Property** : customFileDetails<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>CustomFileDetails(details=>  
details.<br/>Title(false).<br/>Name(true).Size(true).<br/>Status(true).Action(false)) <br/> | **Not  
Applicable** |

| Options to customize File list dialog | **Property**: dialogAction <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>DialogAction(details=>  
details.<br/>Modal(false).<br/>CloseOnComplete(true).<br/>Resize(true).Drag(false).<br/>conte  
nt("#dialogTarget")) <br/> | **Not Applicable** |

| Customize dialog position | **Property**: dialogPosition<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>DialogPosition<br/>{position =>  
position.X(300).Y(100)}<br/> | **Not Applicable** |

| Change file list key values | **Property**: dialogText<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>DialogText(details=> details.<br/>Title(Upload  
File List).Name(File Name).<br/>Size(File Size))<br/> | **Not Applicable** |

| Change drop area text | **Property**: dropAreaText<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>DropAreaText("Drop files here") | No separate  
Property to change dropAreaText. It can be customize using locale Texts |

| Change drop area height | **Property**: dropAreaHeight<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>DropAreaHeight("100%") | Not Applicable |

| Change drop area width | **Property**: dropAreaWidth <br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>DropAreaWidth("100%") | Not Applicable |

| Dynamically push the file | **Property**: pushFile<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>PushFile("files") | Not Applicable |

| Show the files uploader in server already. | **Not Applicable** | **Property:** files <br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Files(@ViewBag.datasource).Render()<br/><br/>public ActionResult PreloadFiles()<br/> { list.Add(new UploaderUploadedFiles<br/> { Name =  
"Nature", Size = 500000, <br/>Type = ".png" }); }<br/> |

| Event triggers when select the file successfully | **Event:** fileSelect <br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
e.FileSelect("fileSelect"))<br/><br/>function fileSelect(e) {}<br/> | **Event:** selected<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Selected("onFileSelect").Render()<br/><br/>function  
onFileSelect(args): void { }<br/> |

## Upload action

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Save URL | **Property** : saveUrl <br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>SaveUrl("Uploadbox/Save") | **Property** :  
saveUrl<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>AsyncSettings(new  
Syncfusion.<br/>EJ2.Inputs.<br/>UploaderAsyncSettings <br/>{SaveUrl =  
@Url.Content("/Uploader/Save"))<br/>.Render() |

| Save URL | **Property** : saveUrl <br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>SaveUrl("Uploadbox/Save") | **Property** :  
saveUrl<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>AsyncSettings(new  
Syncfusion.<br/>EJ2.Inputs.<br/>UploaderAsyncSettings <br/>{SaveUrl =  
@Url.Content("/Uploader/Save"))<br/>.Render() |

| Save URL | **Property** : saveUrl <br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>SaveUrl("Uploadbox/Save") | **Property** :  
saveUrl<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>AsyncSettings(new  
Syncfusion.<br/>EJ2.Inputs.<br/>UploaderAsyncSettings <br/>{SaveUrl =  
@Url.Content("/Uploader/Save") }<br/>.Render() |

| Save URL | **Property** : saveUrl <br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>SaveUrl("Uploadbox/Save") | **Property** :  
saveUrl<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>AsyncSettings(new  
Syncfusion.<br/>EJ2.Inputs.<br/>UploaderAsyncSettings <br/>{SaveUrl =  
@Url.Content("/Uploader/Save"))<br/>.Render() |

| Remove URL | **Property** : removeUrl<br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>RemoveUrl("UploadBox/Remove") | **Property** :  
removeUrl<br/><br/> @Html.EJS().Uploader("UploadFiles").<br/>AsyncSettings(new  
Syncfusion.<br/>EJ2.Inputs.<br/>UploaderAsyncSettings <br/>{RemoveUrl =  
@Url.Content<br/>("/Uploader/Remove"))<br/>.Render() |

| Automatically upload the file when files added in to upload queue | **Property:** autoUpload <br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>AutoUpload(false) | **Property:** autoUpload<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>AutoUpload(false).Render() |

| Synchronous upload | **Property:** asyncUpload<br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>AsyncUpload(false) | No Separate Property for enabling synchronous upload. It can be enabling by using below configuration <br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>AutoUpload(false).Render() |

| Key to get the selected files in server side | **Property:** uploadName<br/><br/>  
@Html.EJ().Uploadbox("UploadBox").<br/>UploadName("Uploadkey") | Id of the element used as key value |

| Upload the files dynamically | **Not Applicable** | Method: upload()<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>AsyncSettings(new  
Syncfusion.EJ2.Inputs.<br/>UploaderAsyncSettings<br/> {SaveUrl =  
@Url.Content<br/>("/Uploader/Save"),<br/> RemoveUrl =  
@Url.Content<br/>("/Uploader/Remove") }).Render()<br/><br/>var uploadobj =  
document.<br/>getElementById("UploadFiles").<br/>ej2\_instances[0]; <br/>uploadobj.upload =  
filesData;<br/> |

| Event triggers before start to upload the action | **Event:** beforeSend<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.OnBeforeSend("onBeforeSend"))<br/><br/>function onBeforeSend(e) {}<br/> | **Event :**  
uploading<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Uploading("beforeUploadStart").<br/>Render()<br/>  
<br/> function beforeUploadStart(args) { }<br/> |

| Event triggers when the upload is in progress | **Event:** inProgress<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.InProgress("uploadInProgress"))<br/><br/>function uploadInProgress(e) {}<br/> | **Event :**  
progress<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Progress("uploadInProgress").<br/>Render()<br/><br/>  
/> function uploadInProgress(args) { }<br/> |

| Event triggers when upload got success | **Event:** success<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Success("uploadSuccess"))<br/><br/>function uploadSuccess(e) {}<br/> | **Event :**  
success<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Success("uploadSuccess").Render()<br/><br/>  
function uploadSuccess(args) { }<br/> |

| Event triggers when upload got failed | **Event:** error<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Error("onUploadError"))<br/><br/>function onUploadError(e) {}<br/> | **Event :**  
failure<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Failure("onUploadFailure").Render()<br/><br/>  
function onUploadFailure(args) { }<br/> |

| Event triggers when the upload got started | **Event:** begin <br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Begin("onUploadBegin"))<br/><br/>function onUploadBegin(e) {}<br/> | **Not Applicable** |

| Event triggers when cancel the upload | **Event:** cancel<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Cancel("onUploadCancel"))<br/><br/>function onUploadCancel(e) {} <br/> | **Event :**  
canceling<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Canceling("uploadingCancel").<br/>Render()<br/><br/>  
/> function uploadingCancel(args) {}<br/> |

| Event triggers when the upload completed | **Event:** complete<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Complete("onUploadComplete"))<br/><br/>function onUploadComplete(e) {}<br/> | **Not**  
**Applicable** |

### Chunk upload

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Enabling the chunk upload | Not Applicable | **Property:** chunkSize<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>MaxFileSize(104857600).AsyncSettings(new  
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =  
@Url.Content("https://aspnetmvc.syncfusion.com/<br/>services/api/uploadbox/Save"),  
<br/>RemoveUrl =  
@Url.Content("https://aspnetmvc.syncfusion.com/<br/>services/api/uploadbox/Remove"),  
ChunkSize = 500000 }).Render()<br/> |

| Retry the upload automatically when it's get failed | Not Applicable | **Property:** retryCount,  
retryAfterDelay<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>MaxFileSize(104857600).AsyncSettings(new  
Syncfusion.EJ2.Inputs.UploaderAsyncSettings { SaveUrl =  
@Url.Content("https://aspnetmvc.syncfusion.com/<br/>services/api/uploadbox/Save"),  
RemoveUrl =  
@Url.Content("https://aspnetmvc.syncfusion.com/<br/>services/api/uploadbox/Remove"),  
ChunkSize = 500000, retryCount = 3, retryAfterDelay = 1000 }).Render()<br/> |

| Pause the uploading file | Not Applicable | **Method:** pause<br/><br/>  
@Html.EJS().Uploader("UploadFiles").Render()<br/><br/>var uploadobj =  
document.<br/>getElementById("UploadFiles").ej2\_instances[0]; <br/>uploadObj.pause =  
filesData;<br/> |

| Event triggers when pausing the file | Not Applicable | **Event:** pausing<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Pausing("onPausingUpload").Render()<br/><br/>func  
tion onPausingUpload(args): void { }<br/> |

| Resuming the paused file | Not Applicable | **Method:** resume<br/><br/>  
@Html.EJS().Uploader("UploadFiles").Render()<br/><br/>var uploadobj =  
document.<br/>getElementById("UploadFiles").ej2\_instances[0];<br/> uploadObj.resume =  
filesData;<br/> |

| Event triggers when resuming the file | Not Applicable | **Event:** resuming<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Resuming("onResumingUpload").Render()<br/><br/>  
function onResumingUpload(args): void { }<br/> |

| Retry the failed file | Not Applicable | **Method:** retry<br/><br/>  
@Html.EJS().Uploader("UploadFiles").Render()<br/><br/>var uploadobj =  
<br/>document.getElementById("UploadFiles").ej2\_instances[0]; <br/>uploadObj.retry =  
filesData;<br/> |

| Cancel the failed file | Not Applicable | **Method:** cancel<br/><br/>  
@Html.EJS().Uploader("UploadFiles").Render()<br/><br/>var uploadobj =  
<br/>document.getElementById("UploadFiles").ej2\_instances[0]; <br/>uploadObj.cancel =  
filesData;<br/> |

| Event triggers when cancel the file | Not Applicable | **Event:** canceling<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Canceling("onCancelingUpload").Render()<br/><br/>  
function onCancelingUpload(args): void { }<br/> |

| Event triggers when chunk file fails | Not Applicable | **Event:** chunkFailure<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>ChunkFailure("onChunkFailure").Render()<br/><br/>  
function onChunkFailure(args): void { }<br/> |

| Event triggers when chunk file success | Not Applicable | **Event:** chunkSuccess<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>ChunkSuccess("onChunkSuccess").Render()<br/><br/>  
> function onChunkSuccess(args): void { }<br/> |

#### Remove action

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Remove the uploaded file | Not Applicable | **Method:** remove<br/><br/>  
@Html.EJS().Uploader("UploadFiles").Render()<br/><br/>var uploadobj =  
document.<br/>getElementById("UploadFiles").ej2\_instances[0]; <br/>uploadObj.remove =  
filesData;<br/> |

| Event triggers when the file removing succeed | **Event:** remove<br/><br/>  
@Html.EJ().<br/>Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
e.Remove("onRemove"))<br/><br/>function onRemove(e) {}<br/> | **Event:** success<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Success("onSuccess").Render()<br/><br/> function  
onSuccess(args) { }<br/> |

| Event triggers when the file removing fails | **Not Applicable** | **Event:** failure<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Failure("onFailure").Render()<br/><br/> function  
onFailure(args) { }<br/> |

#### Buttons

| **Behavior** | **Property in essential JS 1** | **Property in essential JS 2** |

| ----- | ----- | ----- |

| Customize button text | **Property** : buttonText<br/><br/>  
@Html.EJ().<br/>Uploadbox("UploadDefault").<br/>Browse(ButtonText.Browse).<br/>Upload(B



```
uttonText.Upload).
Cancel(ButtonText.Cancel)
 | Property : buttons

@Html.EJS().
Uploader("UploadFiles").
Buttons(ViewBag.UploadButtons).Render()

>
public ActionResult DefaultFunctionalities() { List<UploaderButton> buttons = new
List<UploaderButton>() { }; buttons.Add(new UploaderButton() { browse = "Choose File", clear =
"Clear Files", upload = "Upload Files" }); ViewBag.UploadButtons = buttons; return View();
}public class DefaultButtonModel { public string browse { get; set; } public string clear { get; set;
} public string upload { get; set; } }
 |
```

### Drag and drop

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Enable drag and drop upload | **Property** : allowDragAndDrop<br/><br/>

@Html.EJ().<br/>Uploadbox("UploadDefault").<br/>AllowDragAndDrop(true) | No separate  
Property to disabling drag and drop |

| Set custom drop area | **Not Applicable** | **Property** : dropArea<br/><br/>

@Html.EJS().<br/>Uploader("UploadFiles").<br/>DropArea(".control-fluid").Render() |

### Common

| **Behavior** | **Property in Essential JS 1** | **Property in Essential JS 2** |

| ----- | ----- | ----- |

| Adding custom class to wrapper element | **Property** : cssClass<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>CssClass("Custom-Class") | Not Applicable |

| Enable/Disable the control | **Property** : enabled<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>Enabled(false).Render()<br/> **Method** : enable(),  
disable()<br/> | **Property**: enabled<br/><br/>

@Html.EJS().<br/>Uploader("UploadFiles").<br/>Enabled(false).Render() |

| Set height for uploader | **Property**: height<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>Height("100%") | **Not Applicable** |

| Set width for uploader | **Property**: width<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>Width("100%") | **Not Applicable** |

| Adding HTML attributes | **Property**: htmlAttributes<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>HtmlAttribute(attribute => <br/>attribute.Aria-  
label("Uploadbox"))<br/> | **Not Applicable** |

| Event triggers when control created successfully | **Event**: create<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Create("onCreate"))<br/><br/>function onCreate(e) {}<br/> | **Event**: created<br/><br/>  
@Html.EJS().<br/>Uploader("UploadFiles").<br/>Created("onCreated").Render()<br/><br/>  
function onCreated(args) { }<br/> |

| Event triggers when destroy the control | **Event**: destroy<br/><br/>

@Html.EJ().Uploadbox("UploadDefault").<br/>ClientSideEvents(e =>  
<br/>e.Destroy("onDestroy"))<br/><br/>function onDestroy(e) {}<br/> | **Not Applicable** |

| Keeping the model values in cookies | **Property:** enablePersistence<br/><br/>  
@Html.EJ().Uploadbox("UploadDefault").<br/>EnablePersistence(true) | **Property:**  
enablePersistence<br/><br/>  
@Html.EJS().<br/>Uploader("UploadFiles").<br/>EnablePersistence(false).Render() |  
| Get the selected files data | **Not Applicable** | **Method:** getFilesData<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>.Render().<br/><br/>var uploadobj =  
document.<br/>getElementById("UploadFiles").<br/>ej2\_instances[0];  
<br/>uploadObj.getFilesData();<br/> |  
| Convert bytes in to MB, GB | **Not Applicable** | **Method:** bytesToSize<br/><br/>  
@Html.EJS().Uploader("UploadFiles").<br/>Render().<br/><br/>var uploadobj =  
document.<br/>getElementById("UploadFiles").<br/>ej2\_instances[0];  
<br/>uploadObj.bytesToSize(5000);<br/> |